CS100M/CIS121/EAS121
Introduction to Computer Programming

Spring 2004

Lectures 10,11
MATLAB Functions

Announcements

- Regular time Prelim 1 (last name):
  - A-K in HO B14
  - L-Z in UP B17
- Early intervention
- A3
- Prelim 2 conflicts? Let's wait another week....

Summary/Motivation

- Problems → Solutions
- Programming for automating
- English sentences → Prog. Lang statements
- Data and Actions
- Want
  - less redundancy
  - ease of development
  - more clarity
- Functions (actions)

Using Functions

- compute something
- compute multiple things
- perform task without returning value
- visualization
- how to design your own?

Case Studies

- Number Guessing
  - obtaining input
  - generating random values
- Tic Tac Toe
  - draw board
  - check if player has won
  - check if draw
- Common?
  - some actions seem higher level than simple arithmetic
  - seem to clutter code
  - how to make these customized actions?

Why not scripts?

- Scripts share all variables!

```
% script 1:
name = 'Dani';

% script 2:
name = 'Shagrath';
```

- Want a way of writing portions of code that do not influence each other's variables

Function Syntax

- Syntax

```
function [outputs] = name(inputs)
%H1 comment line accessed by LOOKFOR
%comments accessed by HELP
statements % optional
return     % optional
```

- Example:

```
function randVal = randInt(low,high)
%RANDINT(L,H): Return random int between L and H
%RANDINT takes 2 parameters, LOW and HIGH, and
% returns a random integer LOW <= r <= HIGH
% RANDINT quits if LOW > HIGH

if low > high
     error('low > high in randInt!');
end
randVal = floor(rand*(high-low+1))+floor(low);
```


Function Elements

- signature/header
- H1 line
- Help
- Body
- Return
- Very handy! from MATLAB Help:

Programming and Data Types: M-File Programming: Basic Parts of a Function M-File
Programming and Data Types: M-File Programming: How Functions Work

Basics of Using Our Function

- Looking up:

```
path
lookfor randInt
help randInt
```

- Using:

```
randInt(0,1)
randInt(1,1000)
x = randInt(1,4)
randInt(100,1)
```


How Does It All Work?

- First focus on understanding mechanics
  - name lookup
  - comments
  - inputs
  - execution and local variables (variables in body)
  - return variables and early return
- Next focus on design

Name Lookup

- MATLAB needs to know what you mean
  - Names (tokens!) can be variables, scripts, built-in values, built-in functions, your own functions
  - Use ***path***: predetermined set and order of locations in which to look!
- Rules right out of MATLAB Help for a name:
  - Checks to see if the name is a variable.
  - Checks to see if the name is a subfunction, a MATLAB function that resides in the same M-file as the calling function.
  - Checks to see if the name is a private function.
  - Checks to see if the name is a function on the MATLAB search path. MATLAB uses the first file it encounters with the specified name.

  If you duplicate function names, MATLAB executes the one found first using the above rules.

Inputs, Calling

- Can have zero input arguments

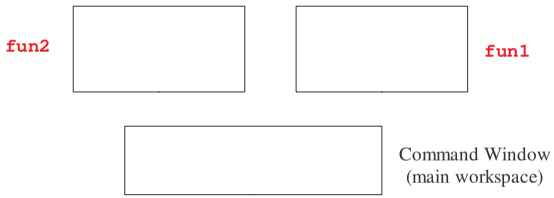
```
function hello
  disp('Hello!');
>> hello
```

- One or more inputs:

```
function hi(prompt)
  disp(prompt);
>> hi
```

Workspace (for params, vars)

- Need a few terms:
  - ***workspace***: portion of MATLAB memory that stores variable values
  - ***scope***: the location/place where a variable's value can be seen/used/changed
- Each function has own workspace



Parameter Passing
Local/Global Variables

- Some more terms:
  - ***caller***: the statement calling a function
  - ***callee***: the function being called
- MATLAB is effectively ***pass by value***:
  - caller copies input values in input variables
  - callee uses input values but cannot change caller's original variables and values (including array contents)

Demo Of Param Passing

```
>> clear
>> add4(1,2)

>> x = 1;
>> y = 2;

>> add4(x,y)

>> x, y

>> result

>> x = 3;
>> add4(x,y)

>> z = add4(x,y)

>> x = add4(x,y)

>> clear
>> add4b(1,2)
```

```
function result = add4(x,y)
   x = x + 2;
   y = y + 2;
   result = x + y;
```

```
function result = add4b(x,y)
   x = x + 2
   y = y + 2
   result = x + y
```


Variables

- Local:
  - input variables, return variables, variables in body
  - seen only in function scope
  - not accessible to other workspaces!
- Global:
  - variable declared to be visible by all workspaces
  - can be dangerous! change once, ruin everything (generally thought to be bad style, but...)
  - useful for constants
  - **`help global`**

Returning from Function

- Can define no return:

```
function hello(prompt)
  disp('Hello!');
```

- Can define one variable to return:

```
function result = iseven(x)
  result = mod(x,2)==0;
```

- Can define multiple variables to return:

```
function [b a] = swap(a, b)
```

  - From command prompt:

```
>> [a b] = swap(1,2)
```


Returning (continued)

- Can return early (new kind of statement):

```
function result = find(target,list)
     result = 0;
     if isempty(list)
          return
     else
       for ii=list
          if ii==target
            result = 1;
            return
          end
       end
     end
```