User-defined Functions

Selim Aksoy
Bilkent University
Department of Computer Engineering
saksoy@cs.bilkent.edu.tr

Scripts

- Command window:
  - x = 2;
  - my_script
    Hello!
  - y = x + 2
    y =
        7
- my_script.m:
  disp('Hello!');
  x = 5;

Scripts

- A script is just a collection of MATLAB statements
- Running a script is the same as running the statements in the command window
- Scripts and the command window share the same set of variables, also called global variables

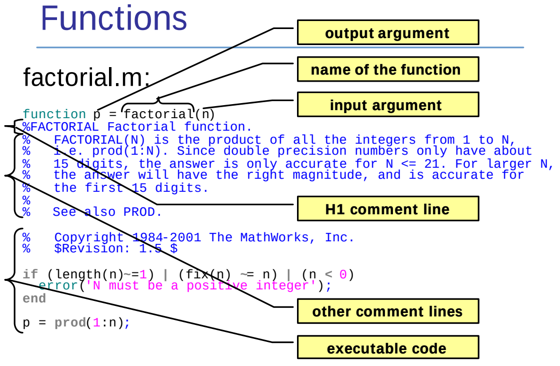
Workspace

- Workspace is the collection of variables that can be used when a command is executing
- Scripts and the command window share the same workspace
- Global variables are problematic because values you depend on may be changed by other scripts

Functions

- A function is a black box that gets some input and produces some output
- We do not care about the inner workings of a function
- Functions provide reusable code
- Functions simplify debugging
- Functions have private workspaces
  - The only variables in the calling program that can be seen by the function are those in the input list
  - The only variables in the function that can be seen by the calling program are those in the output list

Functions



factorial.m:

```
function p = factorial(n)
%FACTORIAL Factorial function.
%   FACTORIAL(N) is the product of all the integers from 1 to N,
%   i.e. prod(1:N). Since double precision numbers only have about
%   15 digits, the answer is only accurate for N <= 21. For larger N,
%   the answer will have the right magnitude, and is accurate for
%   the first 15 digits.
%
%   See also PROD.

%   Copyright 1984-2001 The MathWorks, Inc.
%   $Revision: 1.5 $

if (length(n)~=1) | (fix(n) ~= n) | (n < 0)
  error('N must be a positive integer');
end

p = prod(1:n);
```

Functions

- The function statement marks the beginning of a function
- The name of the function must be the same as the name of the m-file
- The lookfor command searches functions according to the H1 comment line
- The help command displays the comment lines from the H1 line until the first non-comment line

Function Examples

four variables declared as input arguments

```
function distance = dist2(x1, y1, x2, y2)
%DIST2 Calculate the distance between two points
% Function DIST2 calculates the distance between
% two points (x1,y1) and (x2,y2) in a Cartesian
% coordinate system.

% Define variables:
%   x1       -- x-position of point 1
%   y1       -- y-position of point 1
%   x2       -- x-position of point 2
%   y2       -- y-position of point 2
%   distance -- Distance between points

%  Record of revisions:
%      Date       Programmer          Description of change
%      ====       ==========          =====================
%    12/15/98    S. J. Chapman        Original code

% Calculate distance.
distance = sqrt((x2-x1).^2 + (y2-y1).^2);
```

Function Examples

- help dist2

```
 DIST2 Calculate the distance between two points
  Function DIST2 calculates the distance between
  two points (x1,y1) and (x2,y2) in a Cartesian
  coordinate system.
```

- lookfor distance

```
DIST2 Calculate the distance between two points
GFWEIGHT Calculate the minimum distance of a linear…
DISTFCM Distance measure in fuzzy c-mean clustering.
…
```

Function Examples

```
%  Script file: test_dist2.m
%
%  Purpose:
%    This program tests function dist2.
%
%  Record of revisions:
%      Date       Programmer          Description of change
%      ====       ==========          =====================
%    12/15/98    S. J. Chapman        Original code
%
% Define variables:
%   ax     -- x-position of point a
%   ay     -- y-position of point a
%   bx     -- x-position of point b
%   by     -- y-position of point b
%   result -- Distance between the points

% Get input data.
disp('Calculate the distance between two points:');
ax = input('Enter x value of point a:   ');
ay = input('Enter y value of point a:   ');
bx = input('Enter x value of point b:   ');
by = input('Enter y value of point b:   ');

% Evaluate function
result = dist2 (ax, ay, bx, by);

% Write out result.
fprintf('The distance between points a and b is %f\n',result);
```

Function Examples

- clear all
- x1 = 0; y1 = 5;
- whos

```
 Name      Size           Bytes  Class

 x1        1x1                8  double array
 y1        1x1                8  double array

Grand total is 2 elements using 16 bytes
```

- test_dist2

```
Calculate the distance between two points:
Enter x value of point a:   1
Enter y value of point a:   1
Enter x value of point b:   4
Enter y value of point b:   5
The distance between points a and b is 5.000000
```

Function Examples

- whos

```
 Name      Size           Bytes  Class

 ax        1x1                8  double array
 ay        1x1                8  double array
 bx        1x1                8  double array
 by        1x1                8  double array
 result    1x1                8  double array
 x1        1x1                8  double array
 y1        1x1                8  double array

Grand total is 7 elements using 56 bytes
```

- x1

```
x1 =
     0
```

- y1

```
y1 =
     5
```

Function Examples

- Problem: write a function called *strsearch* that takes a string s and a character c, and returns the number of occurrences of c in s and the index of the first occurrence.
- Pseudocode:
  - For each character of s in reverse order
    - If character is equal to c
      - increment the counter
      - save the index

Function Examples

two variables declared as output arguments

```
function [ cnt, pos ] = strsearch( s, c )
%STRSEARCH find the number of occurrences of a character in a string
%   Function STRSEARCH finds the number of occurrences of a character
%   c in a given string s. It returns both the index of the first
%   occurrence and the number of occurrences.
%   It returns 0 for both the index and the number of occurrences if
%   c does not exists in s.
%
%   By Pinar Senkul, 24/10/2003

pos = 0;
cnt = 0;

n = length(s);
for ii = n:-1:1,
    if ( s(ii) == c ),
        cnt = cnt + 1;
        pos = ii;
    end
end
```

Function Examples

- [a, b] = strsearch('abccdecfac', 'c')
  ```
  a =
       4
  b =
       3
  ```
- a = strsearch('abccdecfac', 'c')
  ```
  a =
       4
  ```
- strsearch('abccdecfac', 'c')
  ```
  ans =
       4
  ```

Function Examples

```
function [mag, angle] = polar_value(x,y)
%POLAR_VALUE Converts (x,y) to (r,theta)
% Function POLAR_VALUE converts an input (x,y)
% value into (r,theta), with theta in degrees.

% Check for (0,0) input arguments, and print out
% a warning message.
if x == 0 & y == 0
   msg = 'Both x any y are zero: angle is meaningless!';
   warning(msg);
end

% Now calculate the magnitude.
mag = sqrt(x.^2 + y.^2);

% And calculate angle in degrees.
angle = atan2(y,x) * 180/pi;
```

Function Examples

```
function [avg, med] = mystats(u)
%MYSTATS Find mean and median.
% Function MYSTATS calculates the average and median
% of a data set.

n = length(u);

% Calculate average.
avg = sum(u)/n;

% Calculate median.
w = sort(u);
if rem(n,2) == 1
   med = w((n+1)/2);
else
   med = ( w(n/2) + w(n/2+1) ) / 2;
end
```

Functions: Summary

- Both scripts and functions are saved as m-files
- Functions are special m-files that receive data through input arguments and return results through output arguments
- Scripts are just a collection of MATLAB statements
- Functions are defined by the function statement in the first line
- Scripts use the global workspace but functions have their own local independent workspaces