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;

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

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

- 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

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);
end
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

```matlab
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
',result);
```

### Function Examples

```
clear all
x1 = 0; y1 = 5;
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 =
```

### Function Examples

```
clear all
x1 = 0; y1 = 5;
whos
Name      Size           Bytes  Class
x1         1x1                8  double array
y1         1x1                8  double array
```

### Function Examples

```
% clear all
% x1 = 0; y1 = 5;
% whos
% Name      Size           Bytes  Class
% x1         1x1                8  double array
% y1         1x1                8  double array
```

### Function Examples

```
% clear all
% x1 = 0; y1 = 5;
% whos
% Name      Size           Bytes  Class
% x1         1x1                8  double array
% y1         1x1                8  double array
```

### Function Examples

```
% clear all
% x1 = 0; y1 = 5;
% whos
% Name      Size           Bytes  Class
% x1         1x1                8  double array
% y1         1x1                8  double array
```

### Function Examples

```
% clear all
% x1 = 0; y1 = 5;
% whos
% Name      Size           Bytes  Class
% x1         1x1                8  double array
% y1         1x1                8  double array
```

### Function Examples

```
% clear all
% x1 = 0; y1 = 5;
% whos
% Name      Size           Bytes  Class
% x1         1x1                8  double array
% y1         1x1                8  double array
```

### Function Examples

```
% clear all
% x1 = 0; y1 = 5;
% whos
% Name      Size           Bytes  Class
% x1         1x1                8  double array
% y1         1x1                8  double array
```
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

Pseudocode:

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

```
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.
% It illustrates the use of optional arguments.
% Check for a legal number of input arguments.
msg = nargchk(1,2,nargin);
error(msg);
% If the y argument is missing, set it to 0.
if nargin < 2
    y = 0;
end
% 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);
% If the second output argument is present, calculate
% angle in degrees.
if nargout == 2
    angle = atan2(y,x) * 180 / pi;
end
```

Optional arguments can be checked using:
- `nargchk`: validates number of arguments
- `nargin`: number of input arguments
- `nargout`: number of output arguments

```
function [ m, a ] = polar_value
??? Error using ==> polar_value
Not enough input arguments.
```

```
function [ m, a ] = polar_value( 1, -1, 1 )
??? Error using ==> polar_value
Too many input arguments.
```

```
function [ m, a ] = polar_value( 1, -1 )
m = 1.4142
a = 45
```

```
m = polar_value( 1, -1 )
m = 1.4142
```
Functions: Subfunctions

mystats.m:

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

n = length(u);
avg = mean(u,n);
med = median(u,n);

function a = mean(v,n)
% Subfunction to calculate average.
 a = sum(v)/n;
end

function m = median(v,n)
% Subfunction to calculate median.
 w = sort(v);
 if rem(n,2) == 1
    m = w((n+1)/2);
 else
    m = (w(n/2)+w(n/2+1))/2;
 end
end

mystats.m: main function
mystats can be called by any other
MATLAB function but mean and median
can only be called by other functions in
the same file

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