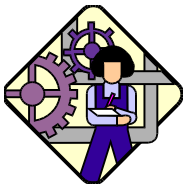
Building your own

Functions



CS112 Scientific Computation
Department of Computer Science
Wellesley College

Built-in MATLAB functions

math: sum, prod, mean, std, abs, sqrt, sin, cos, abs, exp, min, max

logical: any, all, & | ~

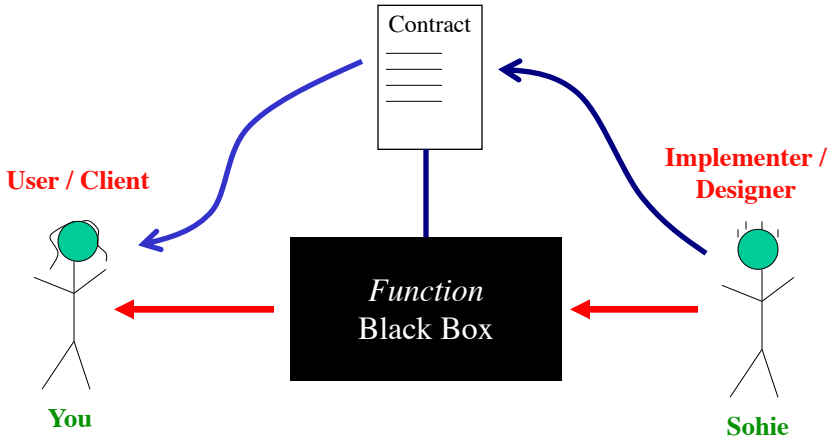
creation: linspace, **:** , ones, zeros

dimensions: size, length

graph/display: plot, figure, subplot, xlabel, ylabel, title, axis, legend, imshow, imtool

input/output: input, disp

Big idea: Abstraction



8-3

The other side of the contract

We've been using functions built by others - let's try writing a few of our own

Implement a **myMean** function that returns a single value representing the average value of a *vector or matrix*

8-4

Rules of the road

output parameter

function name (stored in file myMean.m)

keyword → `function avg = myMean (data)` ← input parameter

```
% avg = myMean(data)
% returns the average of all of the values
% in data, which may be a vector or matrix
dims = size(data);
if (min(dims) == 1)
    avg = sum(data)/length(data);
else
    avg = sum(sum(data))/prod(dims);
end
```

help comments contract

local variable* → `dims`

assign output → `avg = ...`

function body

* Local variables only exist during the execution of the function

Calling the new myMean function

```
>> nums = [3  9  6  2  8];
>> meanVal = myMean(nums)
meanVal =
      5.6000
>> nums = [3  4  7;  2  8  6]
nums =
     3    4    7
     2    8    6
>> meanVal = myMean(nums)
meanVal =
     5.0000
```

Executing a function

MATLAB workspace

Execution land

```
>>
```

8-7

Create nums vector

MATLAB workspace

nums

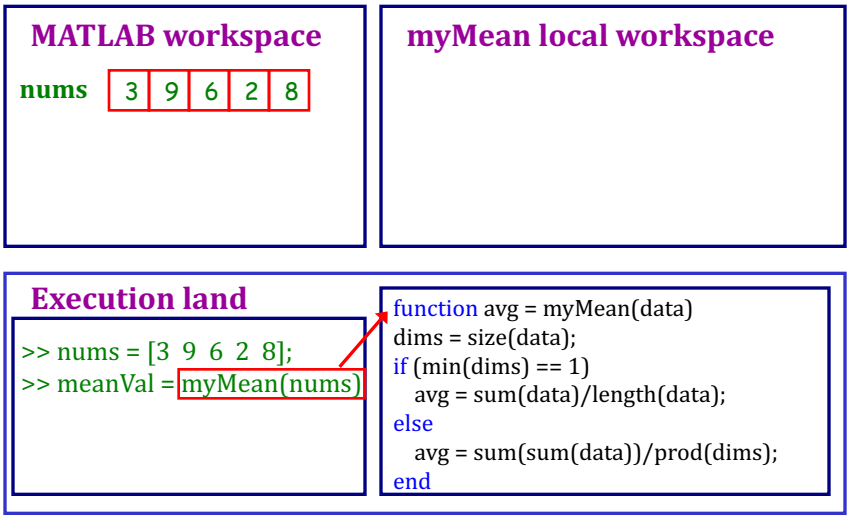
3	9	6	2	8

Execution land

```
>> nums = [3  9  6  2  8];
```

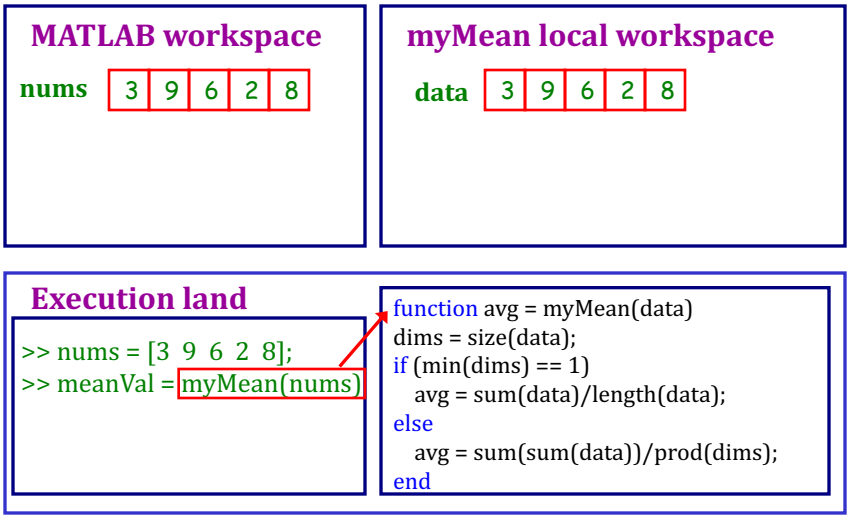
8-8

Invoke myMean function



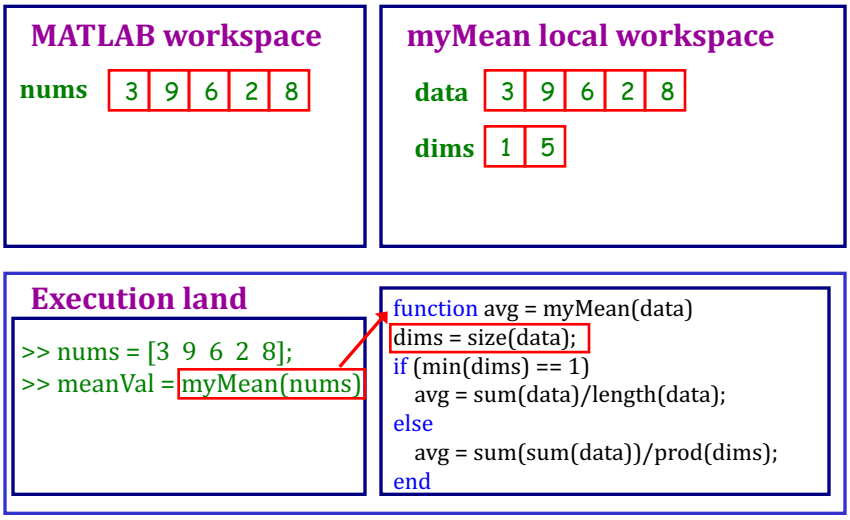
8-9

Create variable for input parameter data and copy value of nums to data



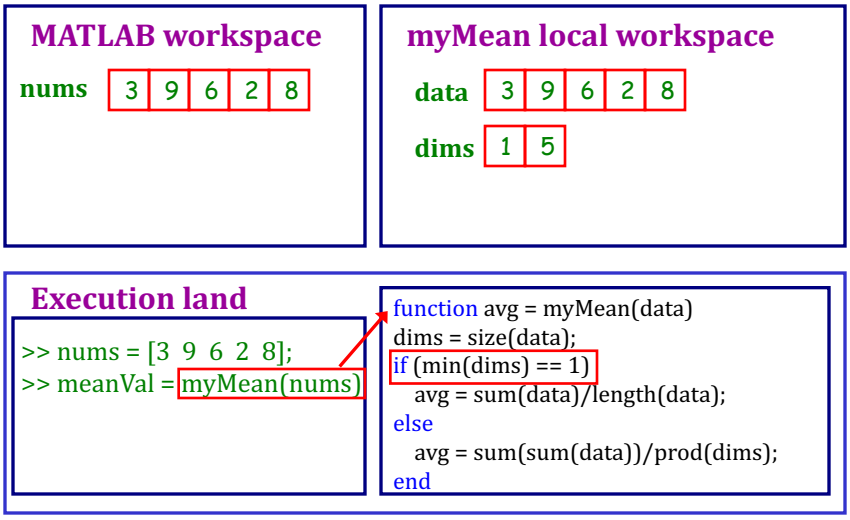
8-10

Execute body of function



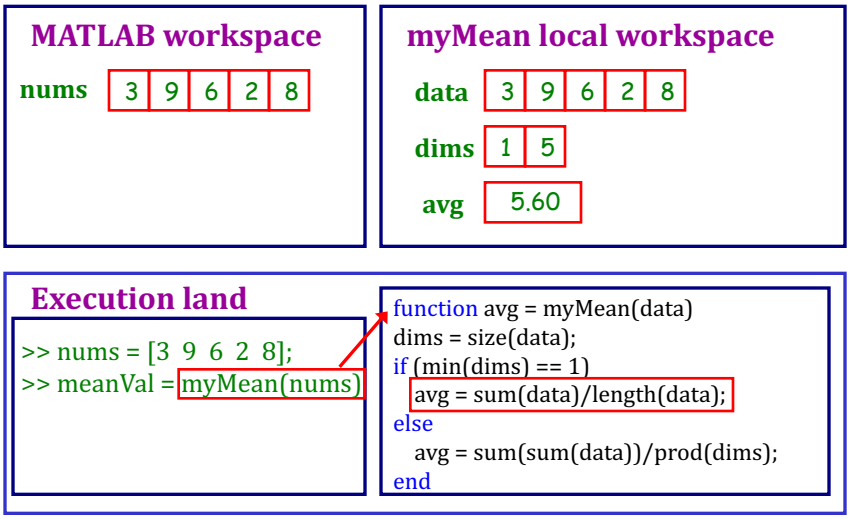
8-11

Is min(dims) == 1?



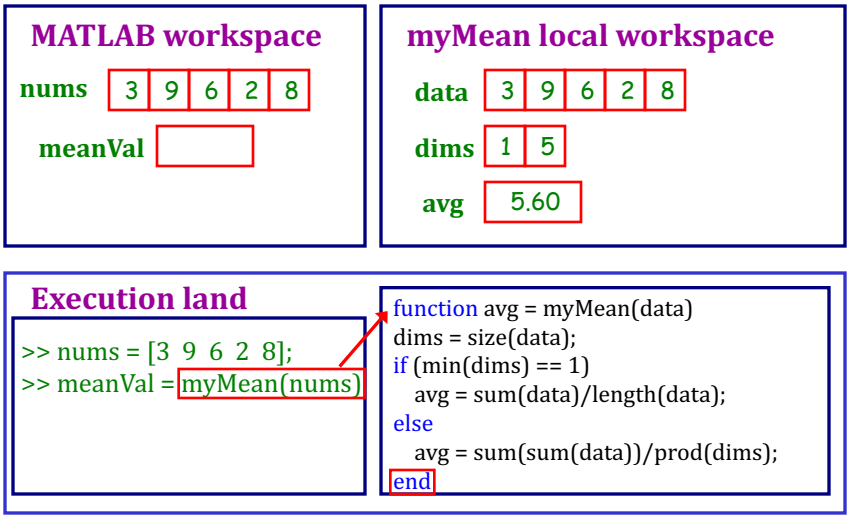
8-12

Yes, so we do 'then' clause



8-13

Return value stored in output variable



8-14

And the local workspace goes away

MATLAB workspace

nums	3	9	6	2	8
meanVal	5.60				

Execution land

```
>> nums = [3  9  6  2  8];
>> meanVal = myMean(nums)
meanVal =
    5.6000
```

8-15

Try another one?

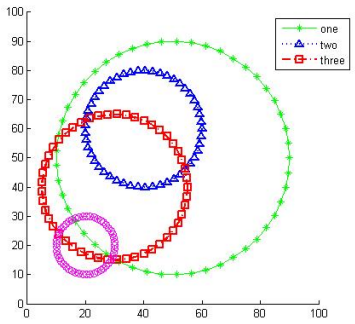
Write a function to draw a circle

Think first about the contract:

use inputs to control appearance: radius, location, color, markers, line style, line width

Call the new function drawCircle

and store it in an M-File named drawCircle.m



8-16

Getting started

```
function drawCircle (radius, xcenter, ycenter, properties, width)
% drawCircle(radius, xcenter, ycenter, properties, width)
% draws circle with the specified radius, centered on location
% (xcenter, ycenter) with the specified properties and width
```

don't forget contract comments!

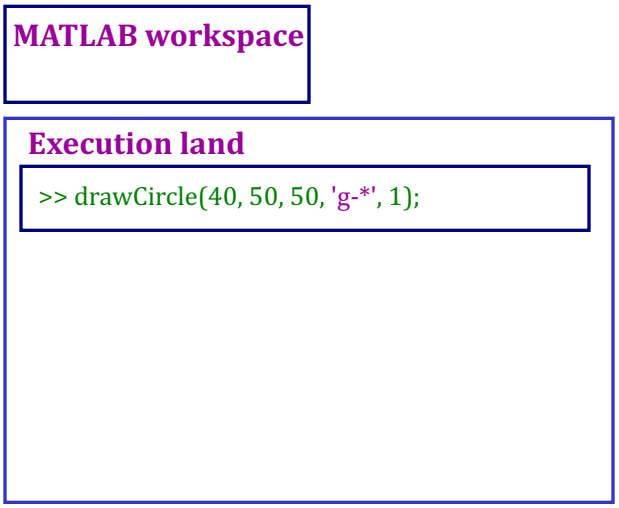
8-17

Filling in the function body

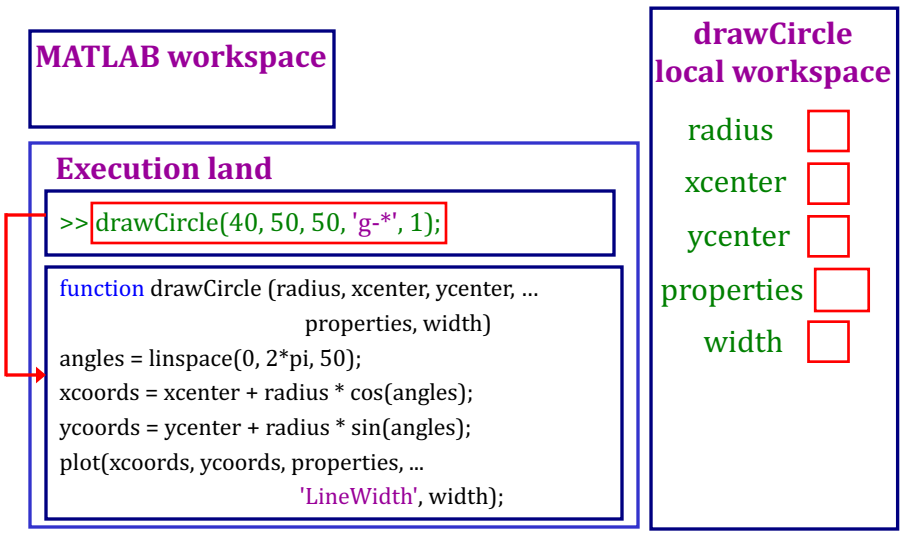
```
function drawCircle (radius, xcenter, ycenter, properties, width)
% drawCircle(radius, xcenter, ycenter, properties, width)
% draws circle with the specified radius, centered on location
% (xcenter, ycenter) with the specified properties and width

angles = linspace(0, 2*pi, 50);
xcoords = xcenter + radius * cos(angles);
ycoords = ycenter + radius * sin(angles);
plot(xcoords, ycoords, properties, 'LineWidth', width);
```

8-18

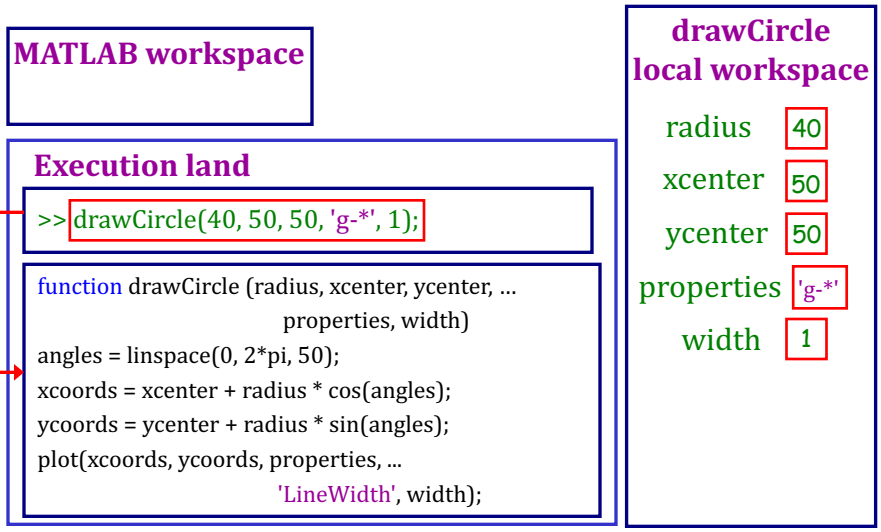
Executing drawCircle function



Create input parameter variables ...

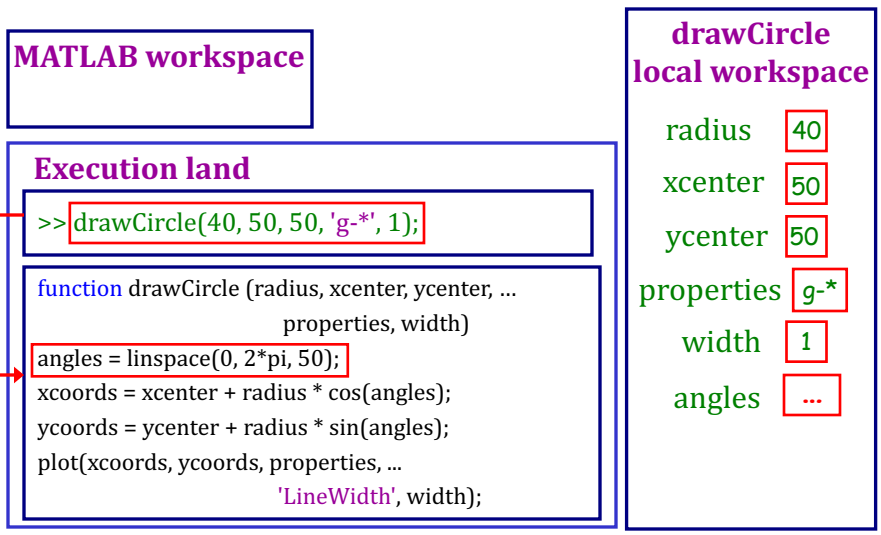


… and fill them in from call statement



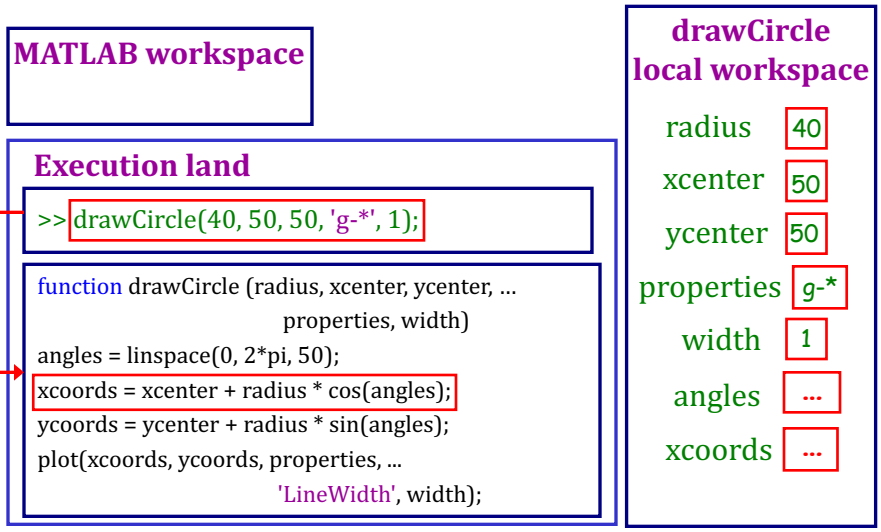
8-21

Execute body of function



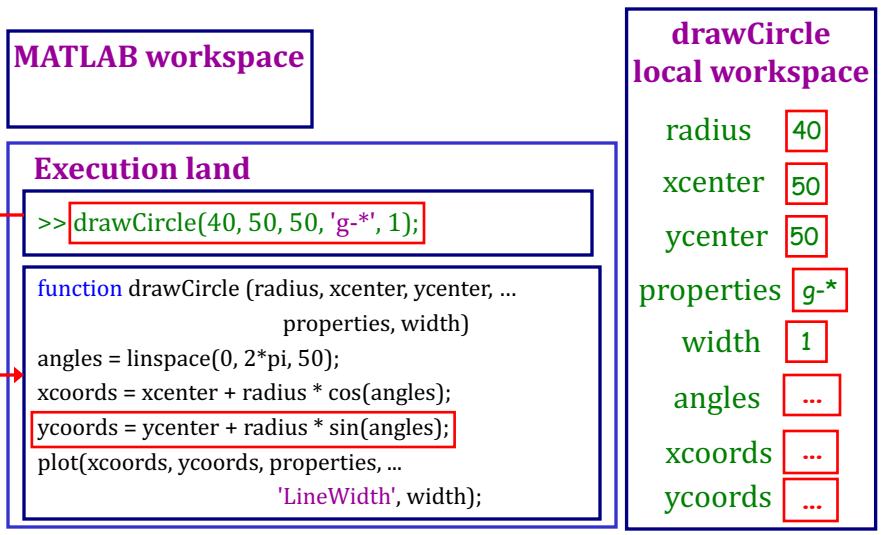
8-22

Next statement



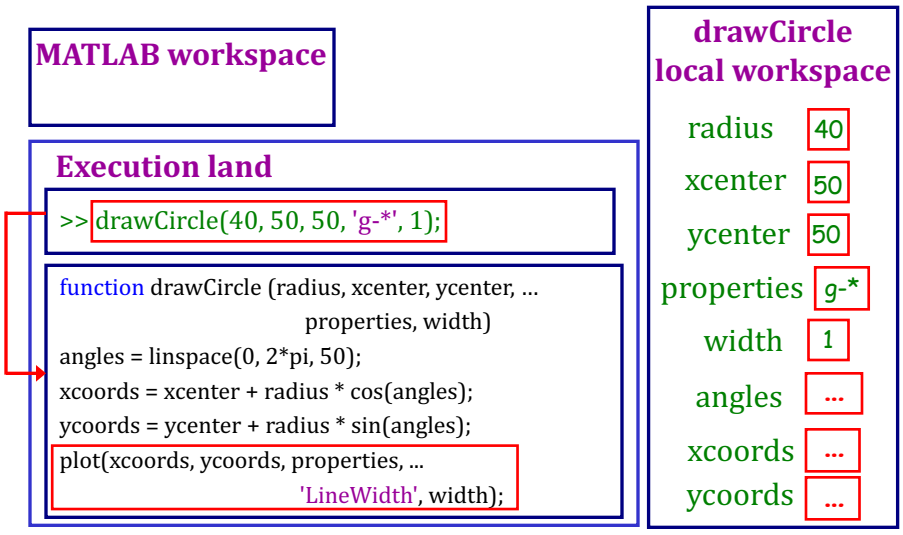
8-23

Next statement



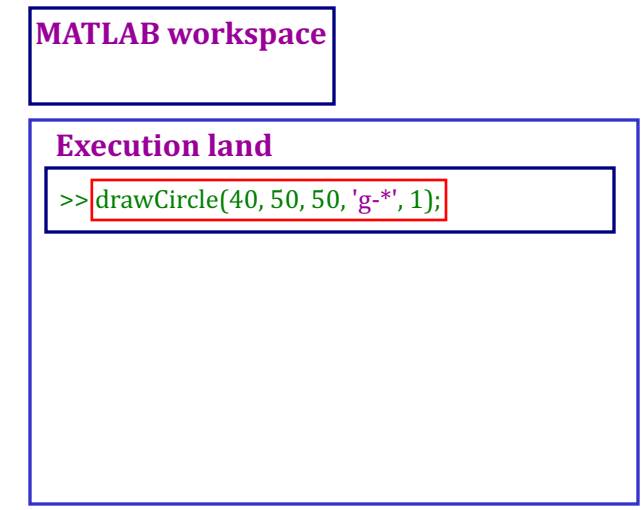
8-24

And we draw the circle



8-25

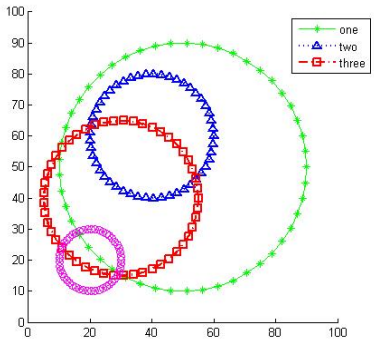
Where'd everybody go?



8-26

A thorough test

```
% testCircle.m
% tests the drawCircle function

hold on
drawCircle(40, 50, 50, 'g-*', 1);
drawCircle(20, 40, 60, 'b:^', 2);
drawCircle(25, 30, 40, 'r-.s', 2);
legend('one',  'two',  'three');
drawCircle(10, 20, 20, 'm--o', 1);
axis equal
axis([0  100  0  100])
hold off
```



Functions with multiple outputs

Suppose we'd like to write a variation of myMean that returns both the *arithmetic mean* and *geometric mean*

Given n numbers:

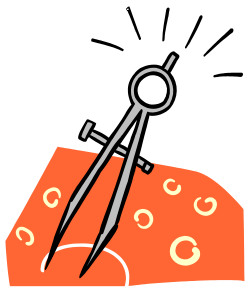
$a_1, a_2 \ldots a_n$

arithmetic mean:

$(a_1 + a_2 + \ldots + a_n)/n$

geometric mean:

$\sqrt[n]{a_1 * a_2 * \ldots * a_n}$



Consider first ...

Some built-in functions can return one or more values, depending on how the function is called

For example, consider the min function:

```
>> nums = [3   9   6   2   8];
>> minVal = min(nums)
minVal =
      2
>> [minVal minIndex] = min(nums)
minVal =
      2
minIndex =
      4
```

8-29

New lean mean machine

```
function [arith geom] = myMean2(data)
% [arith geom] = myMean2(data)
% returns both the arithmetic and geometric mean of
% the values in data, which may be a vector or matrix

dims = size(data);
if (min(dims) == 1)
   arith = sum(data)/length(data);
   geom = nthroot(prod(data), length(data));
else
   arith = sum(sum(data))/prod(dims);
   geom = nthroot(prod(prod(data)), prod(dims));
end
```

8-30