# Introduction to Matlab

Weichung Wang

2003 NCTS-NSF Workshop on Differential Equations,
Surface Theory, and Mathematical Visualization

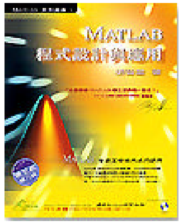
NCTS, Hsinchu, February 13, 2003

# Main Goals

- ✍A brief Matlab introduction for beginners
- ✍Warm-up for Prof. Karcher’s courses

# Tutorials

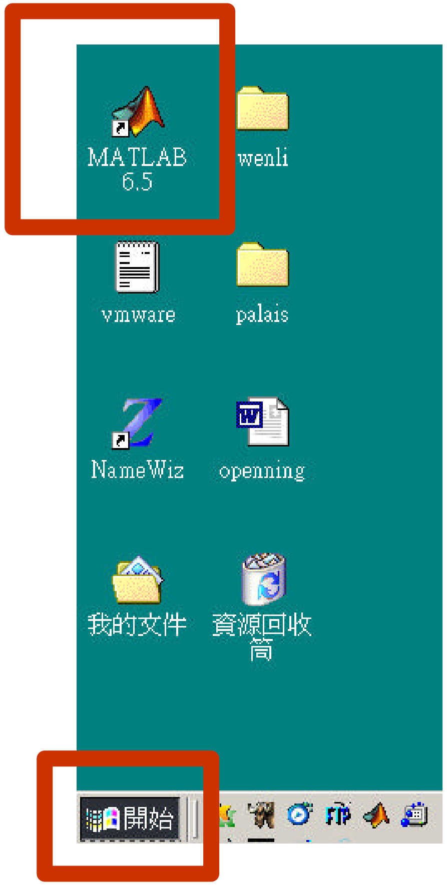
- MATLAB **程式設計與應用，** **張智星，初版，清蔚科技**, 2000

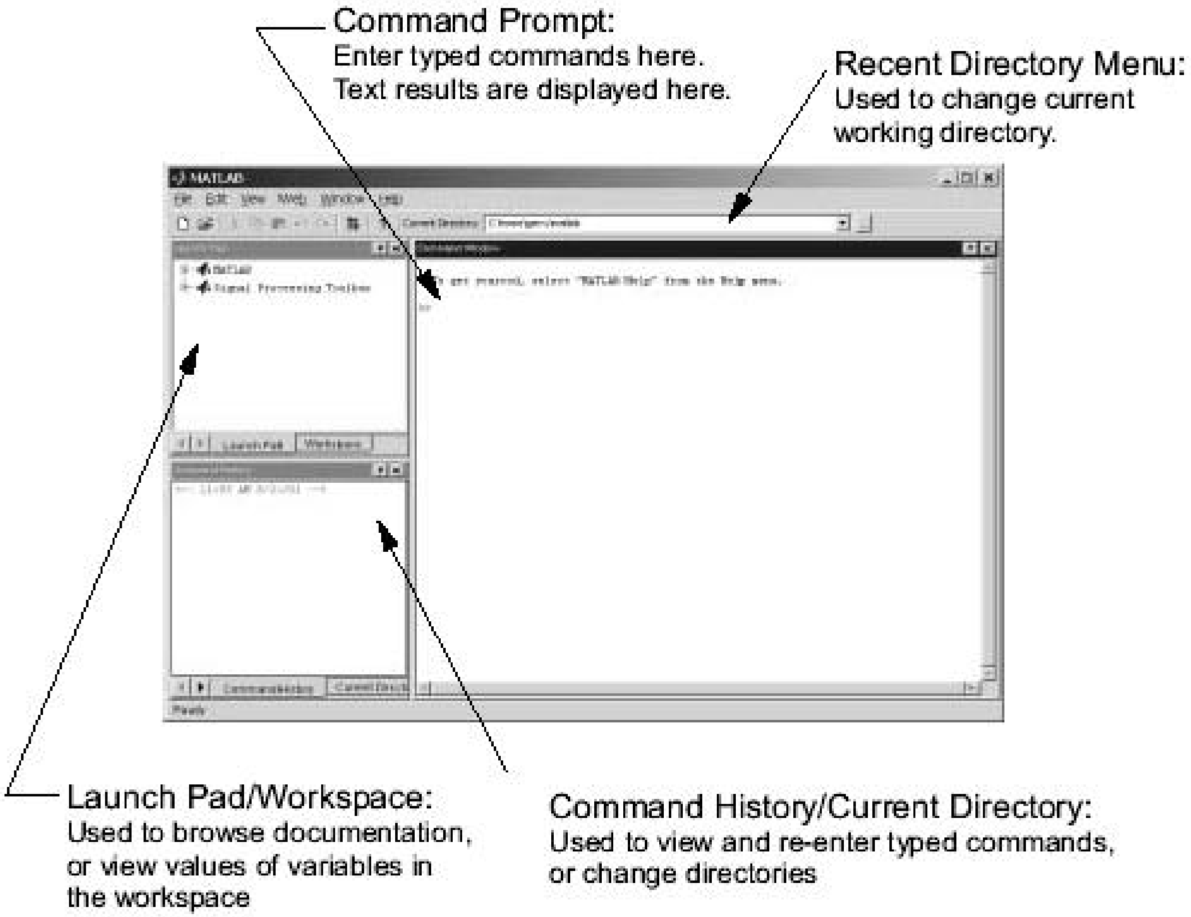


- Slides by Gerald Recktenwald http://www.prenhall.com/recktenwald.

- Many online tutorial on the Internet http://www.mines.utah.edu/gg_computer_seminar/matlab/matlab.html

# Start Matlab in Windows





Command Prompt:
Enter typed commands here.
Text results are displayed here.

Recent Directory Menu:
Used to change current
working directory.

Launch Pad/Workspace:
Used to browse documentation,
or view values of variables in
the workspace

Command History/Current Directory:
Used to view and re-enter typed commands,
or change directories

# Start Matlab in Unix shell

- ✍In command line window, type

```
>matlab
>matlab -nojvm
```

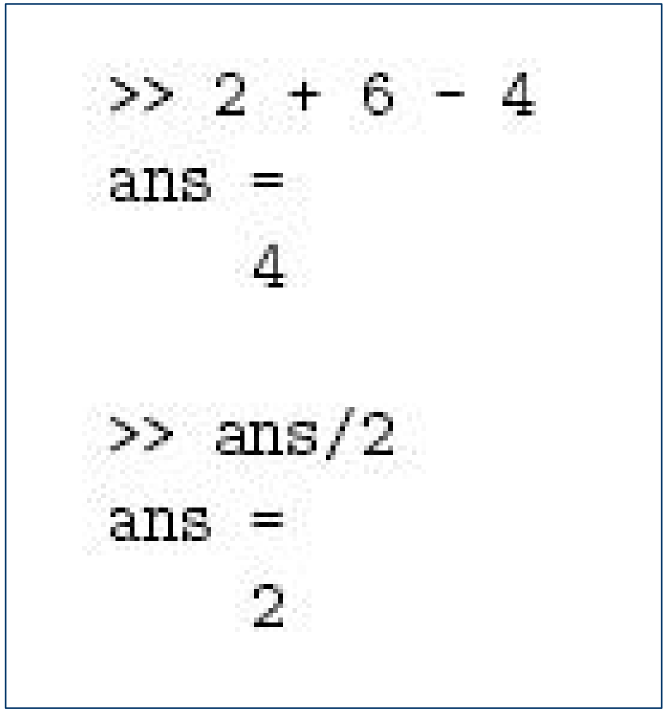
# What is Matlab?

- MATLAB == MATrix LABoratory
- Mathworks: http://www.mathworks.com
- Major software characteristics:
  - matrix-based numeric computation
  - high-level programming language
  - graphics & visualization
  - toolboxes provide application-specific functionality

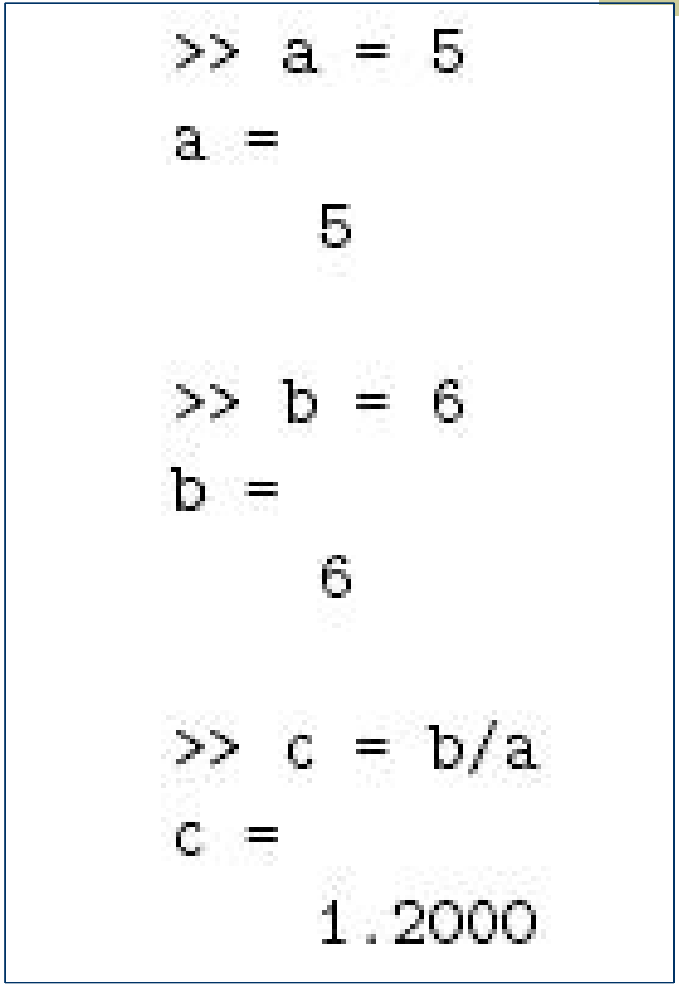
# What Is Matlab? (Cont.)

- ✍Multi-platform support (PC / Macintosh / UNIX)
- ✍Interfaces to other systems.
  - ✍Custom C, Fortran (**MATLAB** is callable)
  - ✍Extensive data I/O facility
- ✍Matlab is case sensitive (`mtxA ~= MTXA`)

# As a Calculator

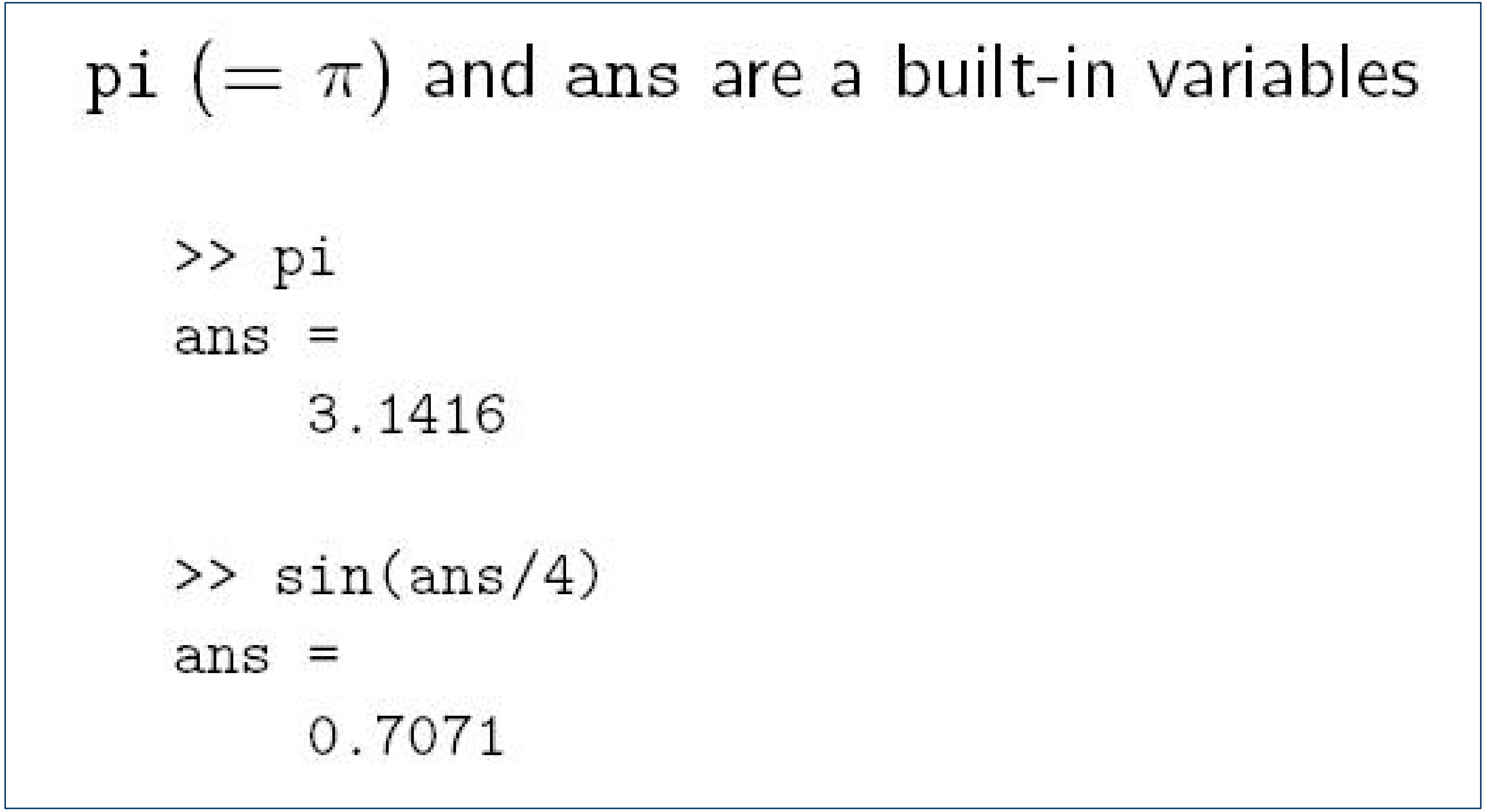


```
>> 2 + 6 - 4
ans =
     4

>> ans/2
ans =
     2
```



```
>> a = 5
a =
     5

>> b = 6
b =
     6

>> c = b/a
c =
     1.2000
```

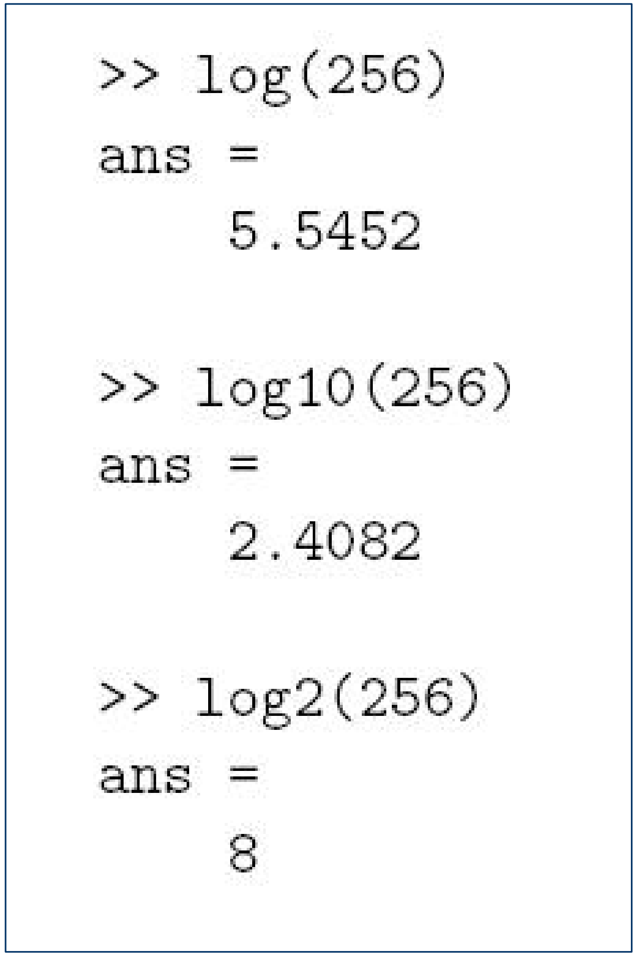
# Built-in Variables



pi ($= \pi$) and ans are a built-in variables

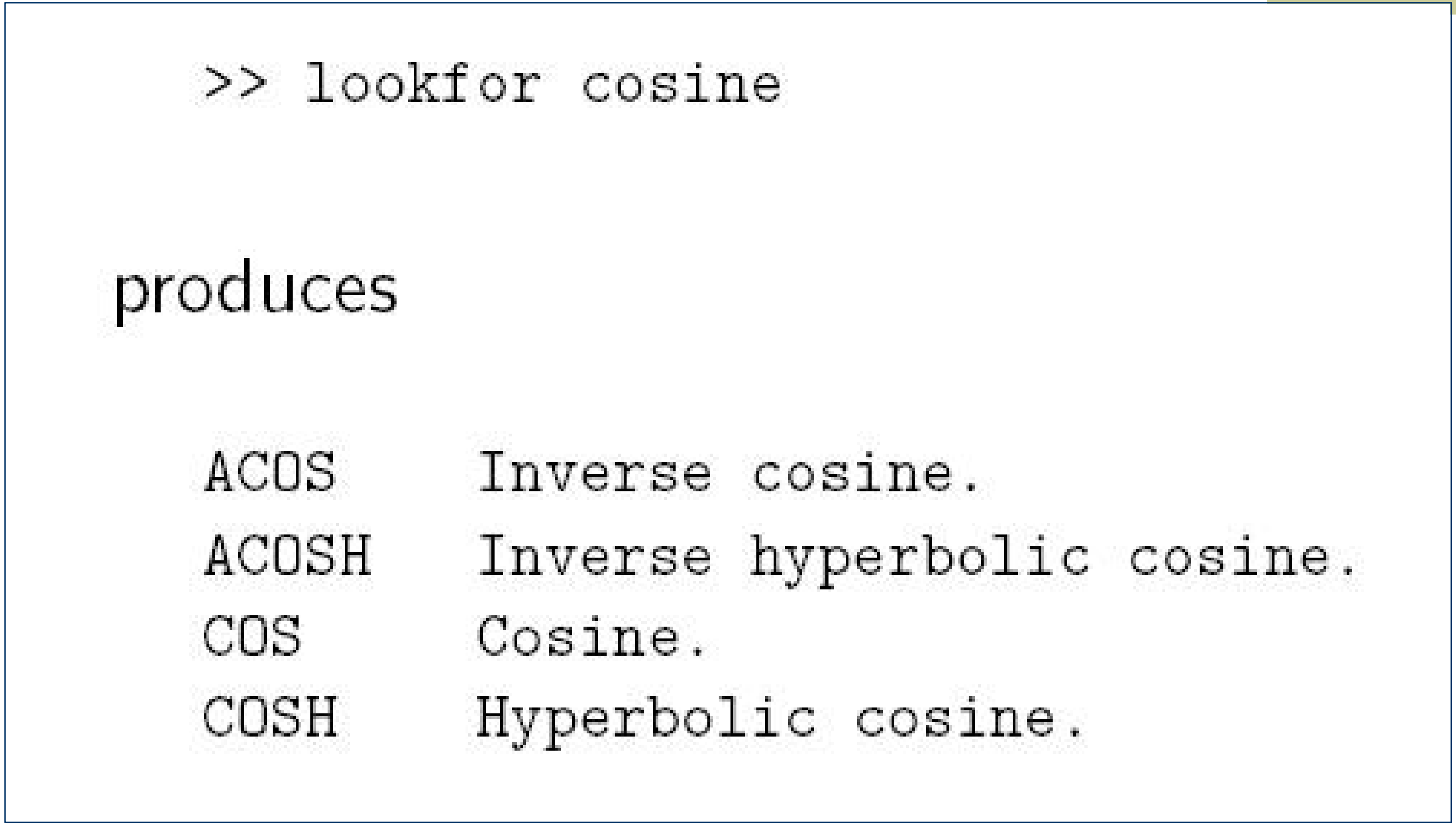
```
>> pi
ans =
     3.1416

>> sin(ans/4)
ans =
     0.7071
```

# Built-in Functions



```
>> log(256)
ans =
     5.5452

>> log10(256)
ans =
     2.4082

>> log2(256)
ans =
     8
```

# Looking for Functions

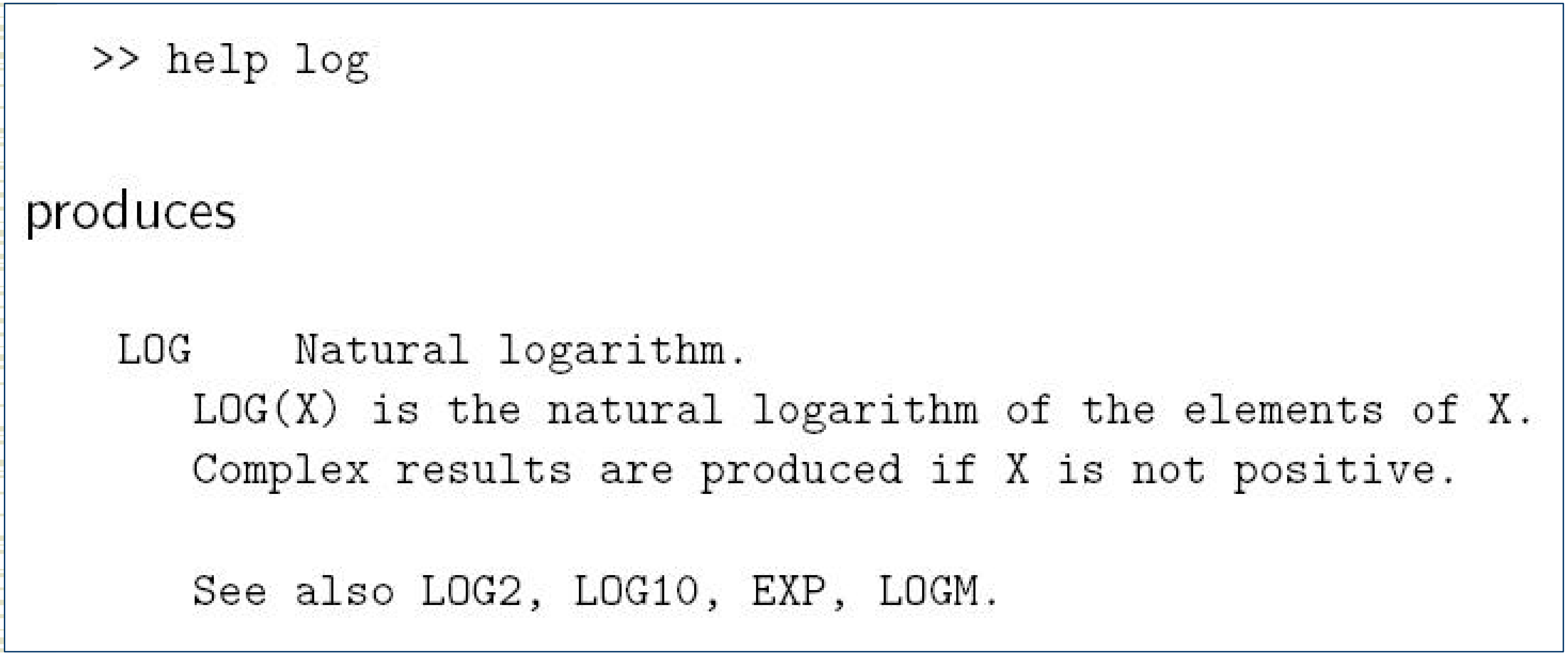


```
>> lookfor cosine
```

produces

```
ACOS     Inverse cosine.
ACOSH    Inverse hyperbolic cosine.
COS      Cosine.
COSH     Hyperbolic cosine.
```

# On-line Help

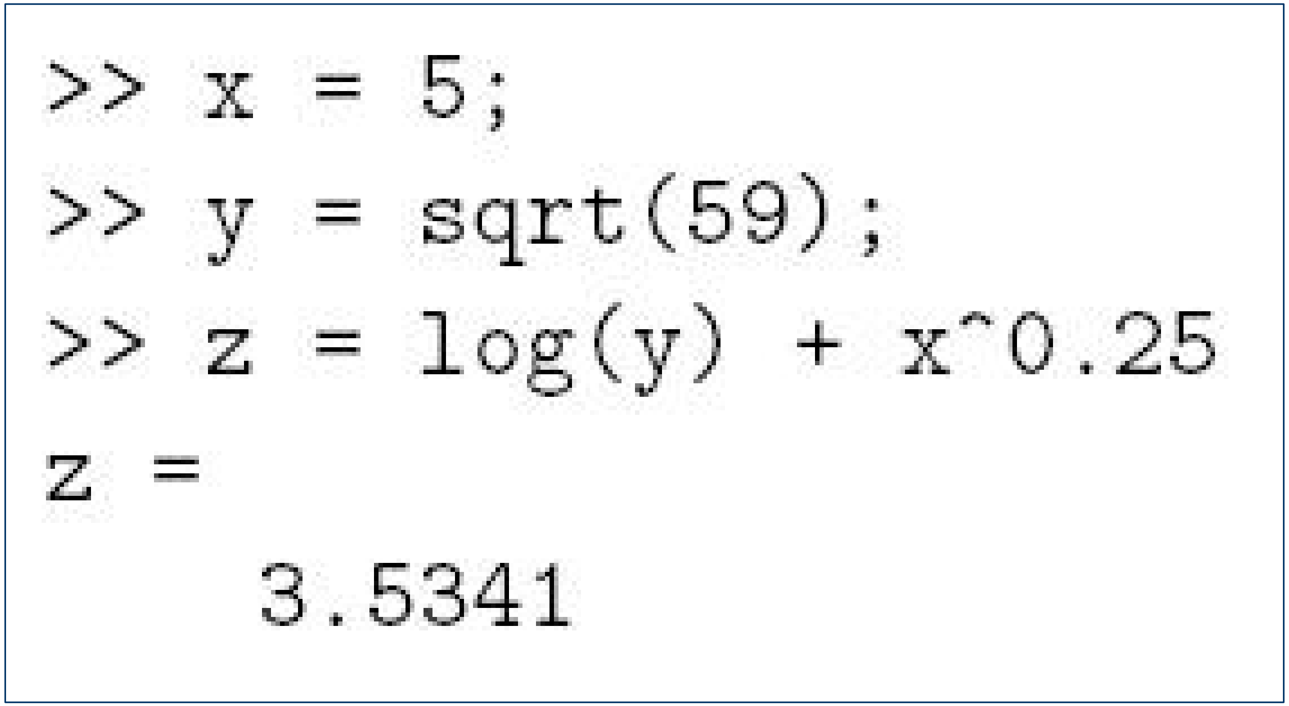


```
>> help log
```

produces

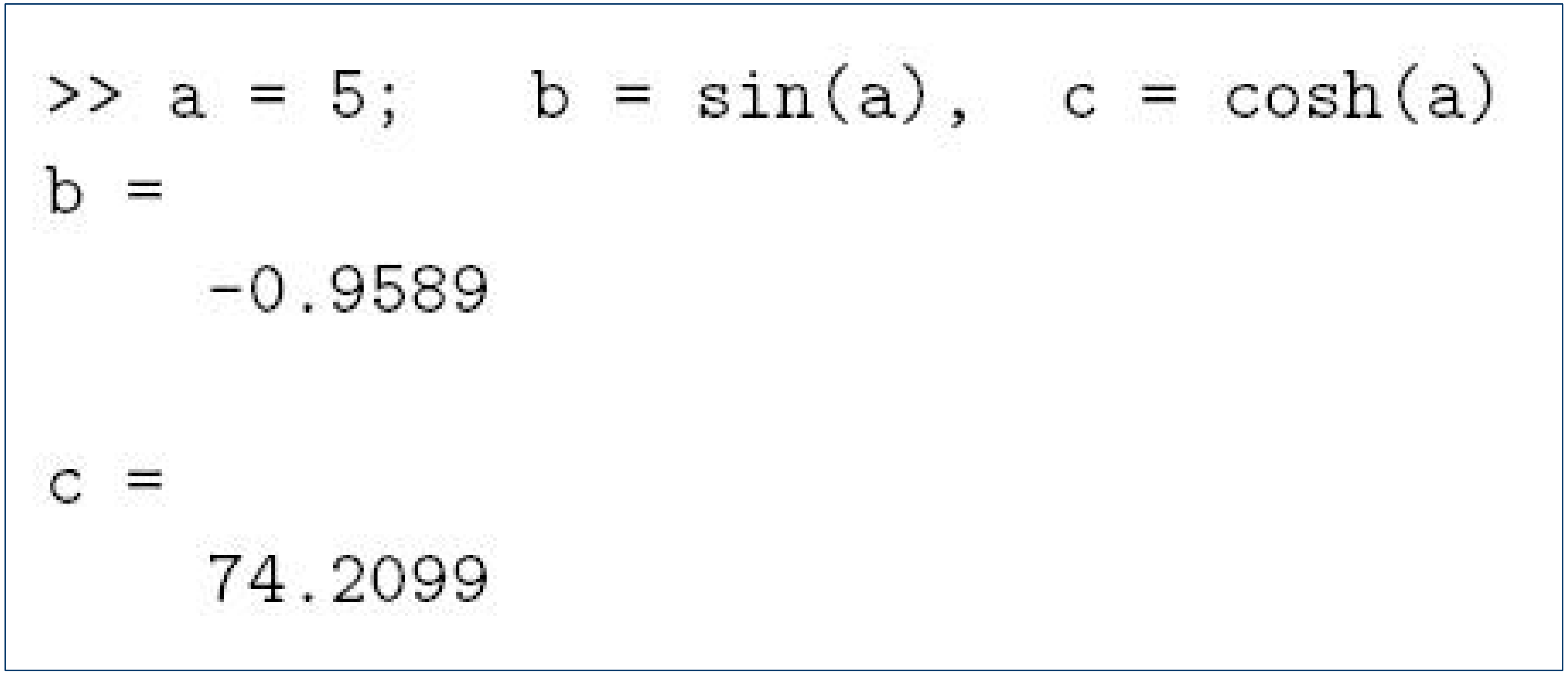
```
LOG    Natural logarithm.
   LOG(X) is the natural logarithm of the elements of X.
   Complex results are produced if X is not positive.

   See also LOG2, LOG10, EXP, LOGM.
```

# Suppress Output With Semicolon



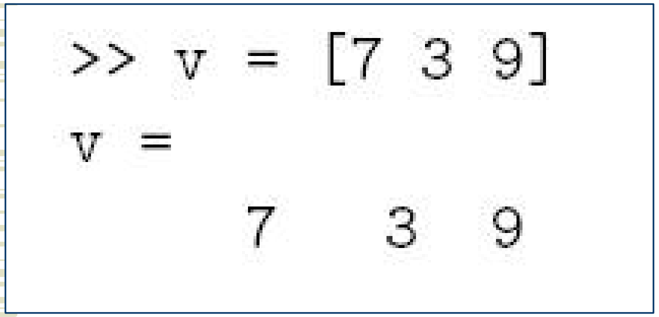
```
>> x = 5;
>> y = sqrt(59);
>> z = log(y) + x^0.25
z =

    3.5341
```

# Multiple Statements Per Line

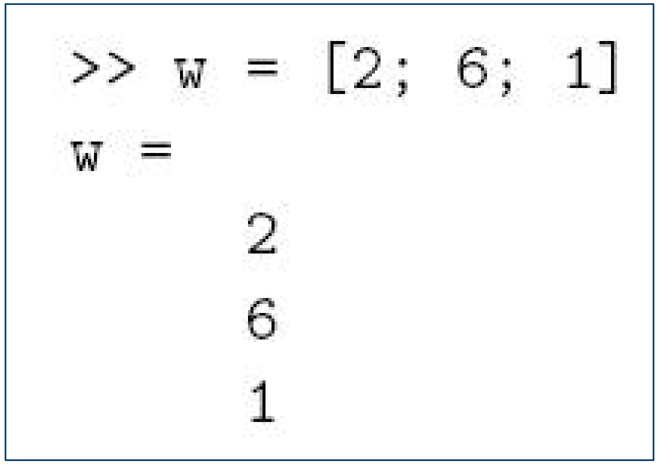


```
>> a = 5;   b = sin(a),  c = cosh(a)
b =
    -0.9589


c =
    74.2099
```

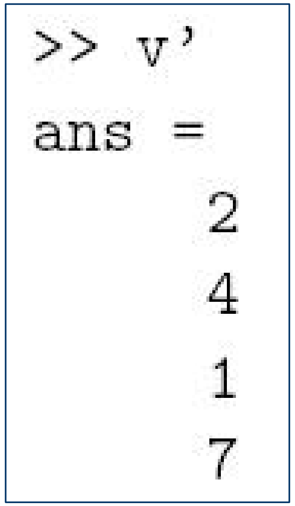
# Vectors and Matrices



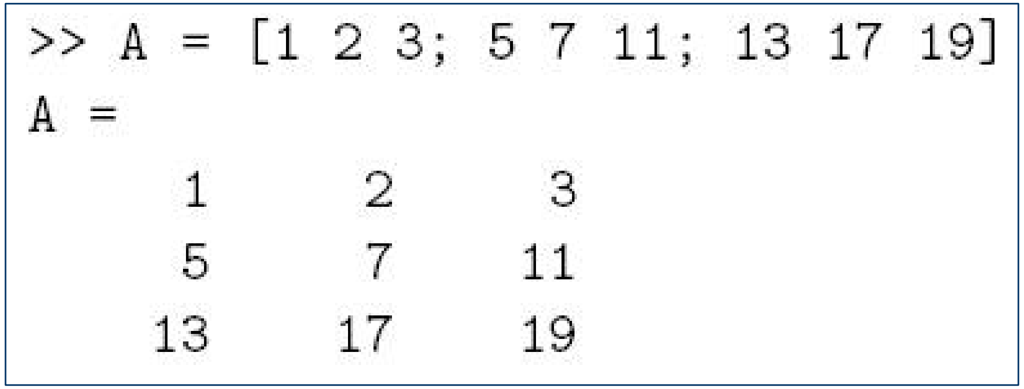
```
>> v = [7 3 9]
v =
      7    3   9
```



```
>> w = [2; 6; 1]
w =
        2
        6
        1
```

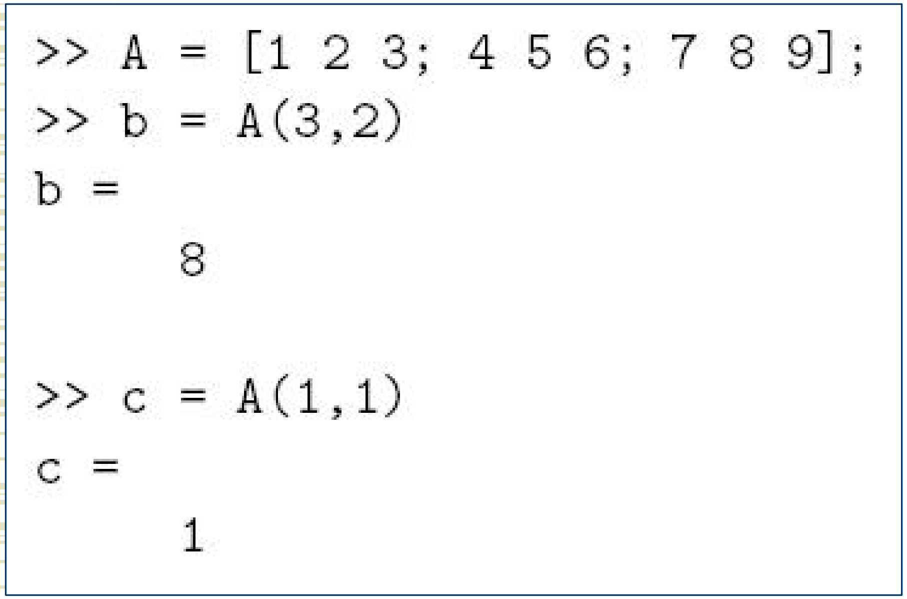


```
>> v'
ans =
      2
      4
      1
      7
```

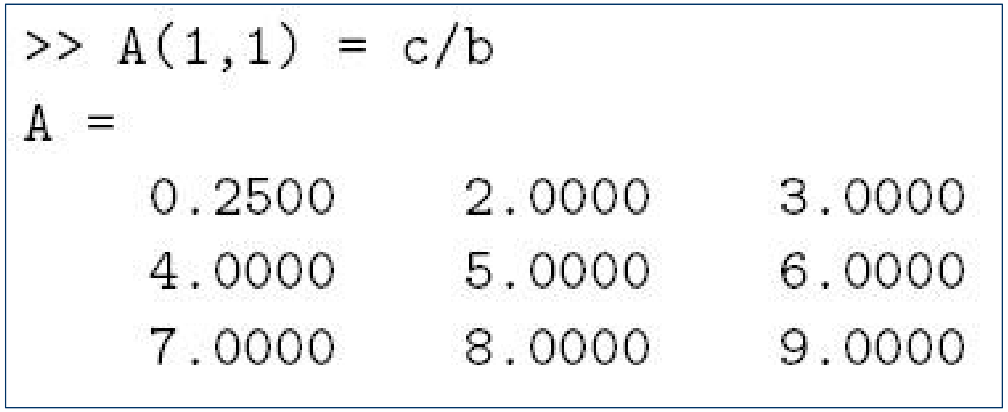


```
>> A = [1 2 3; 5 7 11; 13 17 19]
A =
       1      2      3
       5      7     11
      13     17     19
```

# Matrix

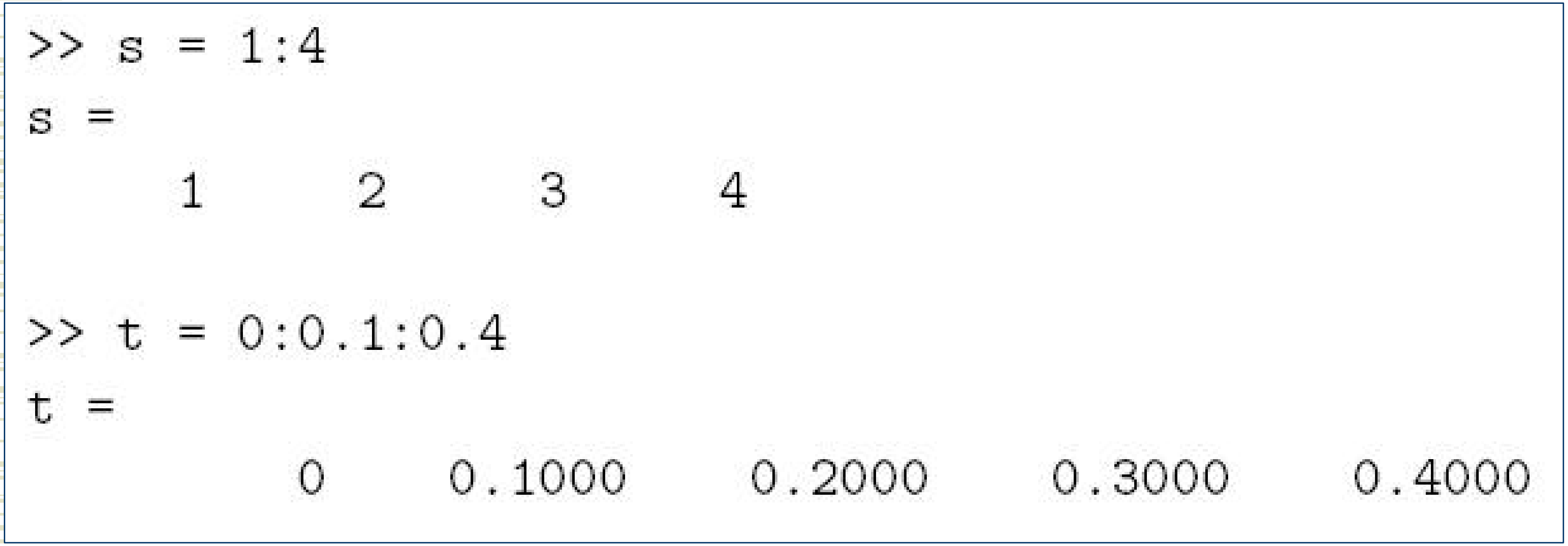


```
>> A = [1 2 3; 4 5 6; 7 8 9];
>> b = A(3,2)
b =
     8

>> c = A(1,1)
c =
     1
```



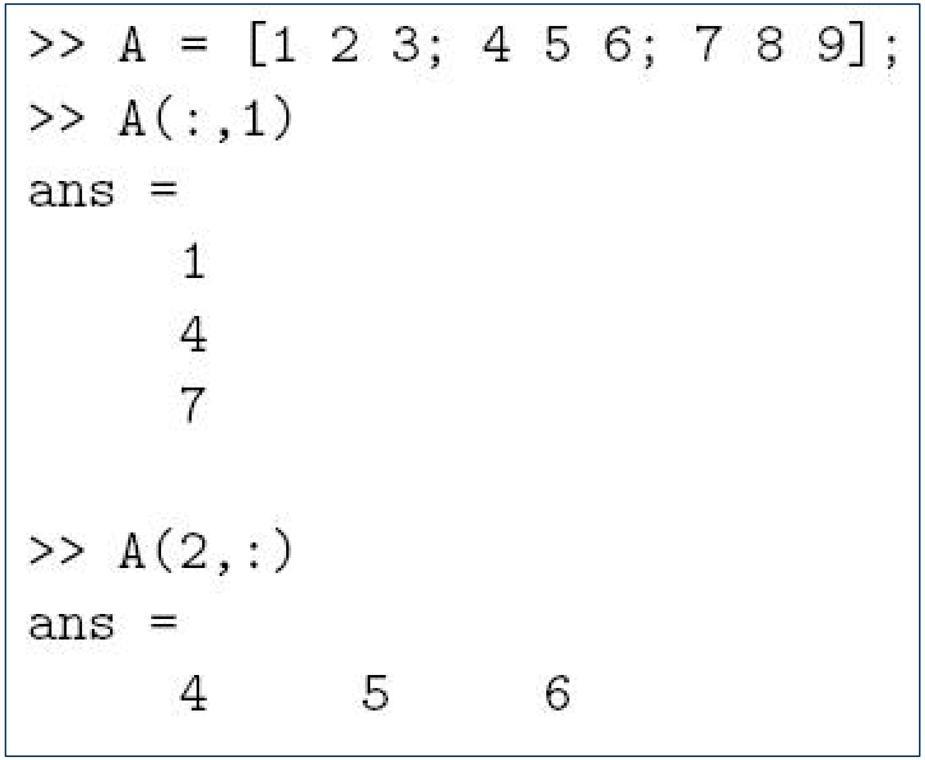
```
>> A(1,1) = c/b
A =
    0.2500    2.0000    3.0000
    4.0000    5.0000    6.0000
    7.0000    8.0000    9.0000
```

# Colon Notation

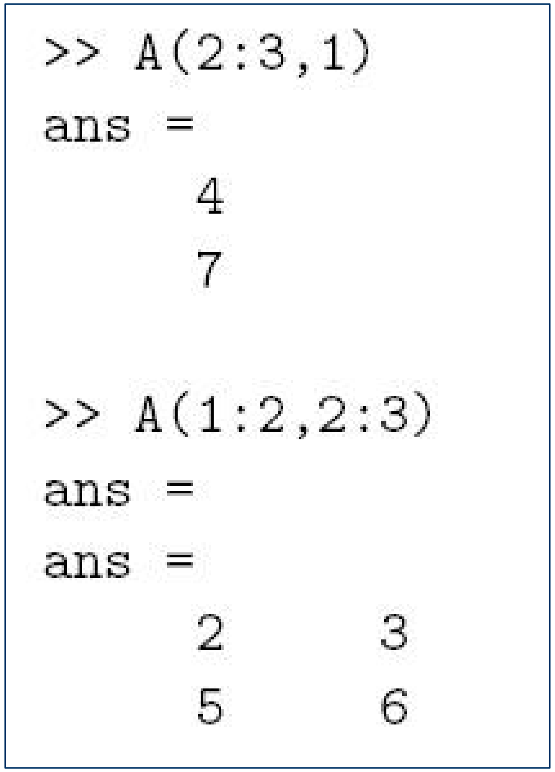


```
>> s = 1:4
s =
     1     2     3     4

>> t = 0:0.1:0.4
t =
         0    0.1000    0.2000    0.3000    0.4000
```

# Colon Notation (Cont.)

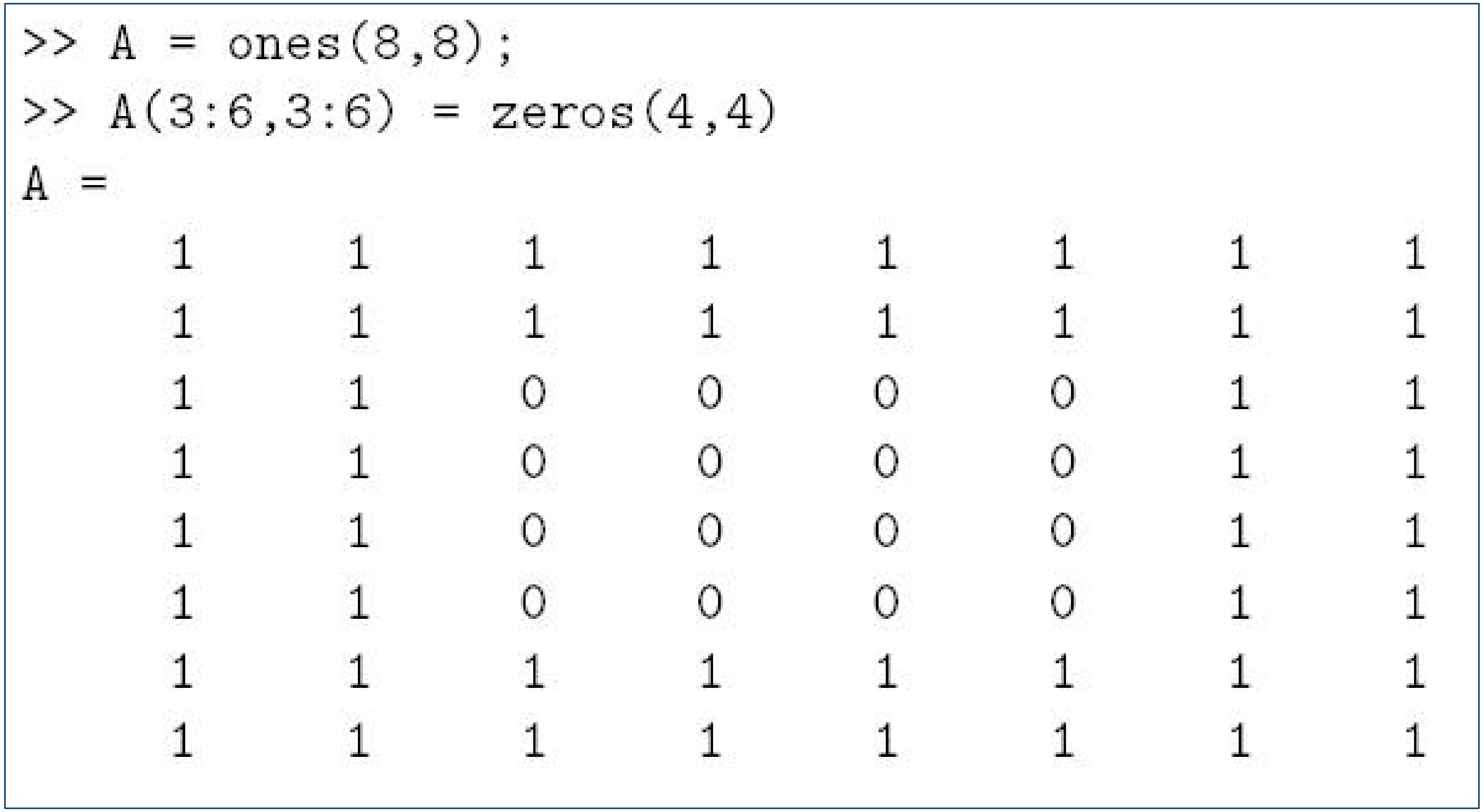


```
>> A = [1 2 3; 4 5 6; 7 8 9];
>> A(:,1)
ans =
     1
     4
     7

>> A(2,:)
ans =
     4     5     6
```



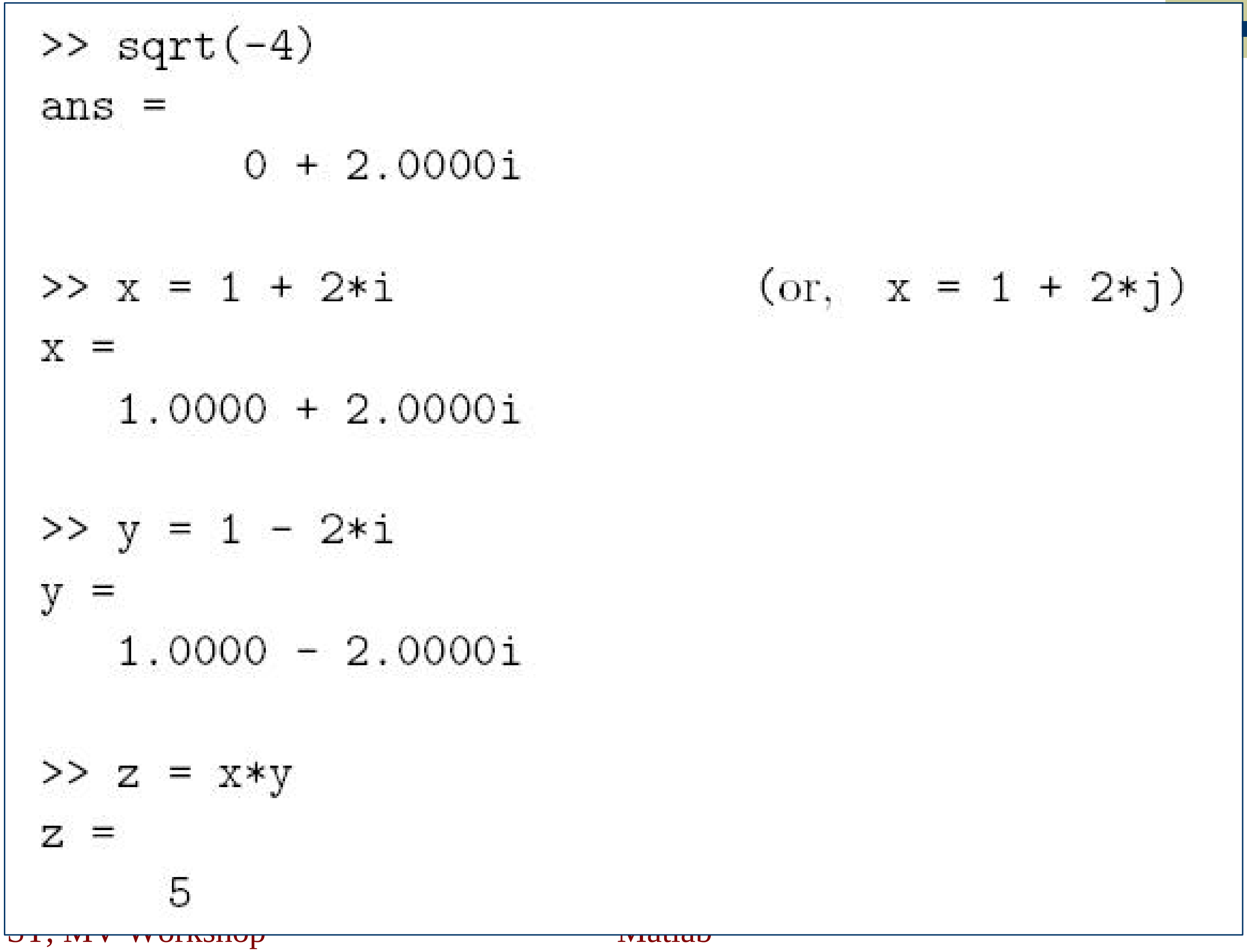
```
>> A(2:3,1)
ans =
     4
     7

>> A(1:2,2:3)
ans =
ans =
     2     3
     5     6
```

# Colon Notation (Cont.)



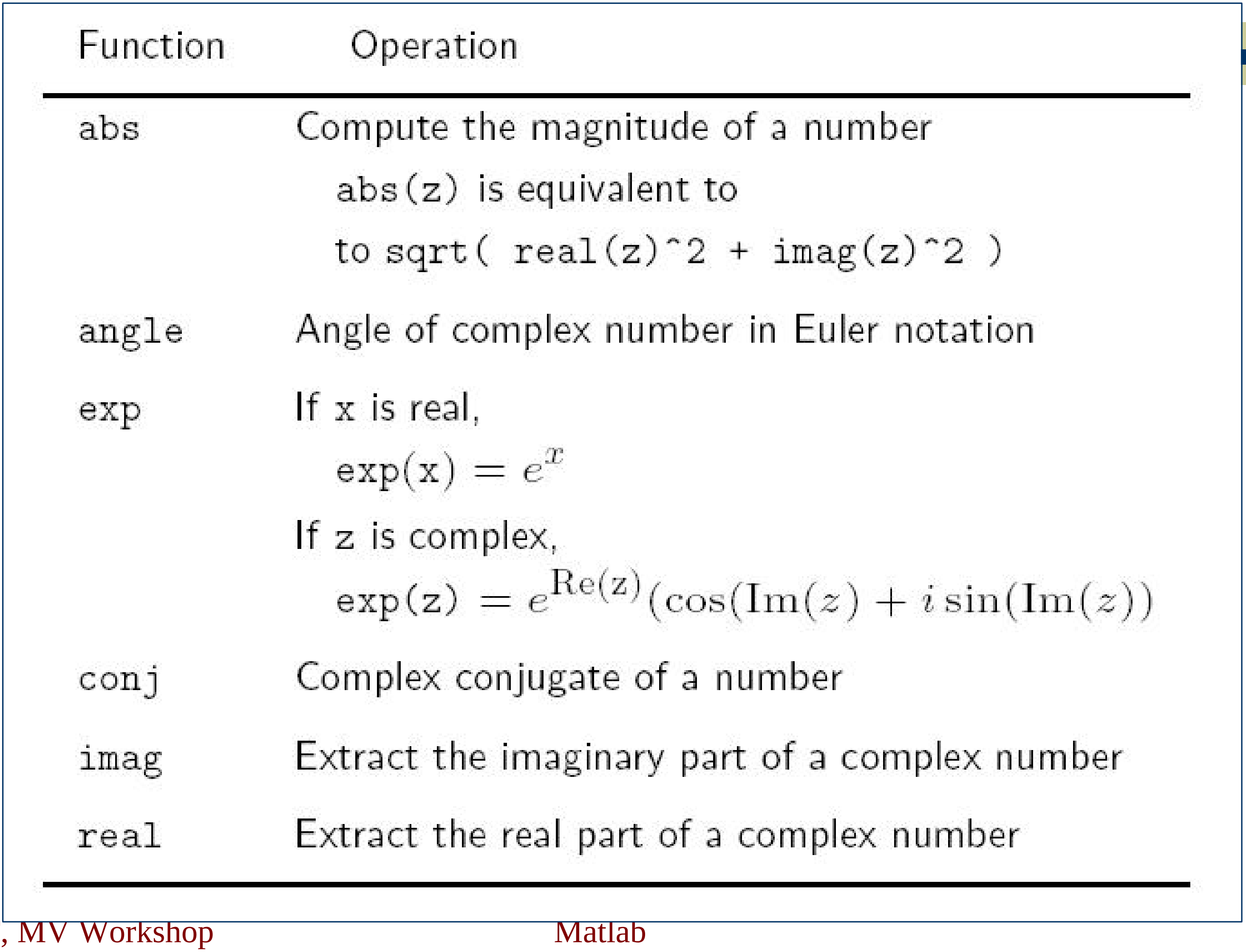
```
>> A = ones(8,8);
>> A(3:6,3:6) = zeros(4,4)
A =
     1     1     1     1     1     1     1     1
     1     1     1     1     1     1     1     1
     1     1     0     0     0     0     1     1
     1     1     0     0     0     0     1     1
     1     1     0     0     0     0     1     1
     1     1     0     0     0     0     1     1
     1     1     1     1     1     1     1     1
     1     1     1     1     1     1     1     1
```

# Complex Numbers



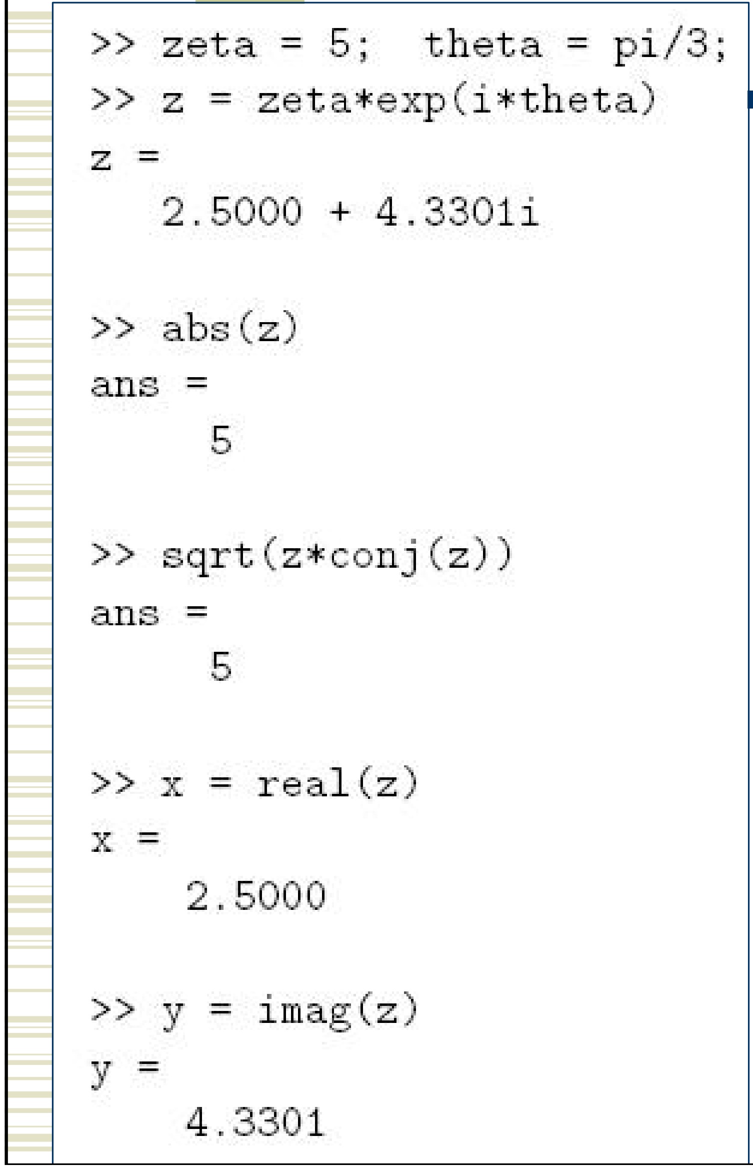
```
>> sqrt(-4)
ans =
        0 + 2.0000i

>> x = 1 + 2*i                    (or,  x = 1 + 2*j)
x =
   1.0000 + 2.0000i

>> y = 1 - 2*i
y =
   1.0000 - 2.0000i

>> z = x*y
z =
     5
```

# Complex Number (Cont.)



| Function | Operation |
|---|---|
| abs | Compute the magnitude of a number<br>abs(z) is equivalent to<br>to sqrt( real(z)^2 + imag(z)^2 ) |
| angle | Angle of complex number in Euler notation |
| exp | If x is real,<br>exp(x) $= e^{x}$<br>If z is complex,<br>exp(z) $= e^{\mathrm{Re(z)}}(\cos(\mathrm{Im}(z) + i\sin(\mathrm{Im}(z))$ |
| conj | Complex conjugate of a number |
| imag | Extract the imaginary part of a complex number |
| real | Extract the real part of a complex number |

# Complex Number (Cont.)

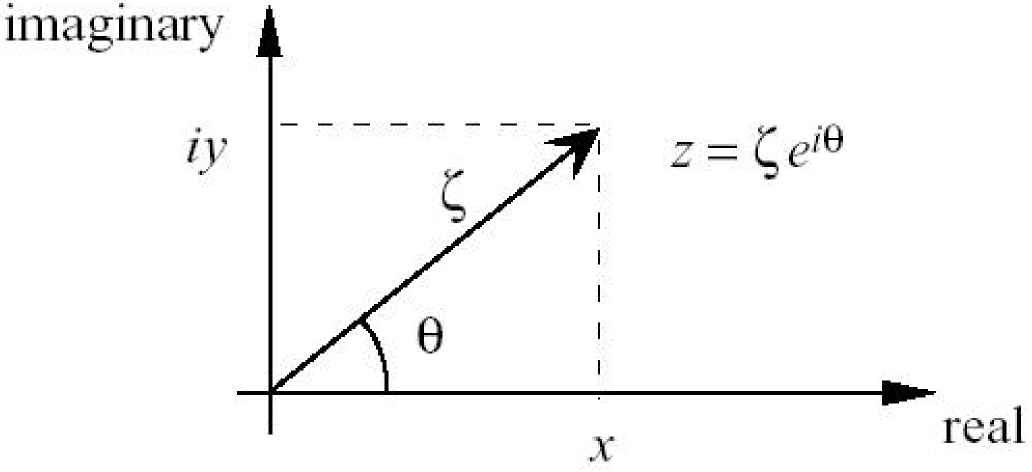


```
>> zeta = 5;  theta = pi/3;
>> z = zeta*exp(i*theta)
z =
   2.5000 + 4.3301i

>> abs(z)
ans =
     5

>> sqrt(z*conj(z))
ans =
     5

>> x = real(z)
x =
    2.5000

>> y = imag(z)
y =
    4.3301
```

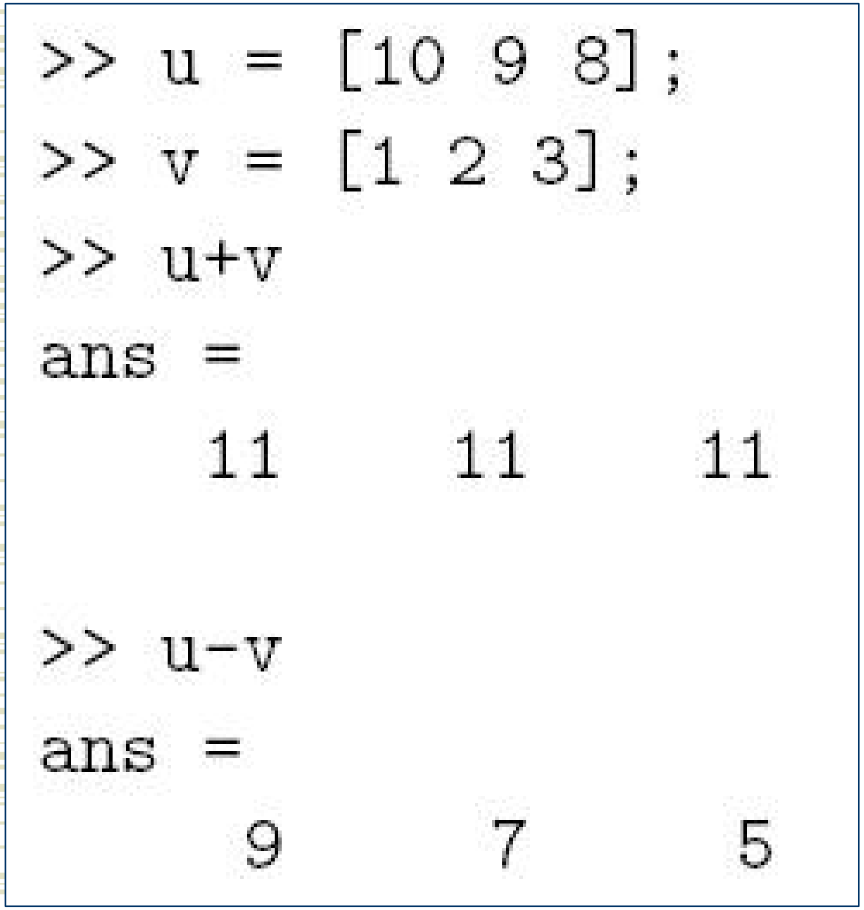
$$z = \zeta e^{i\theta}$$

$$x = \mathrm{Re}(z) = |z|\cos(\theta) = \zeta\cos(\theta)$$

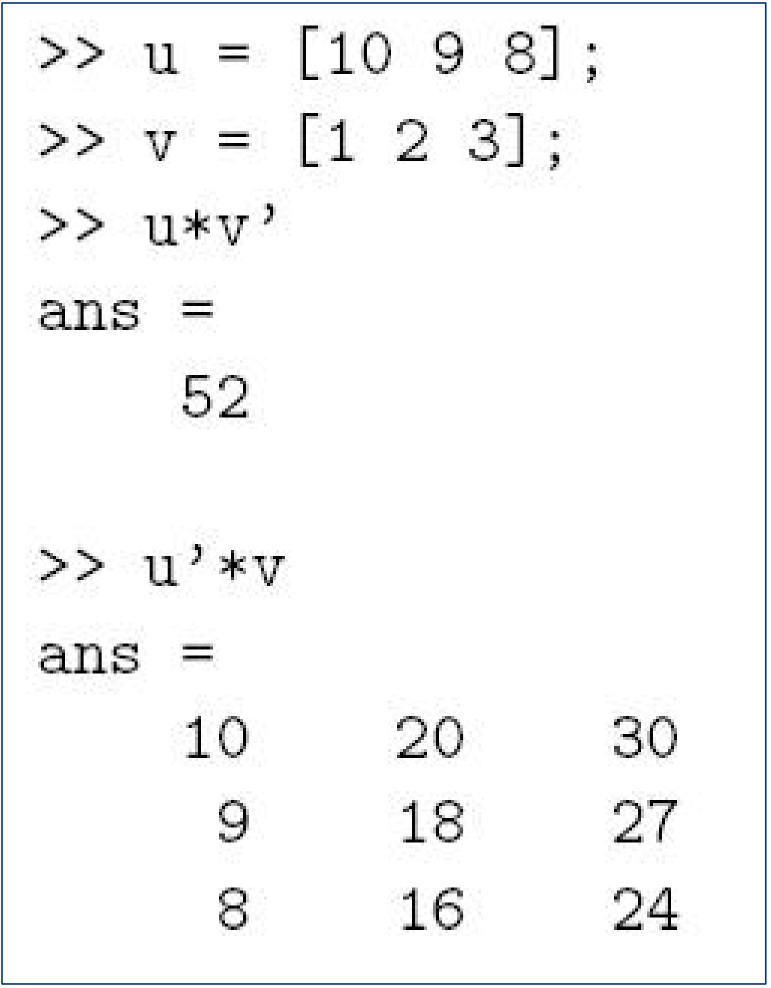
$$y = i\mathrm{Im}(z) = i|z|\sin(\theta) = i\zeta\sin(\theta)$$



# Manipulation of Matrices and Vectors

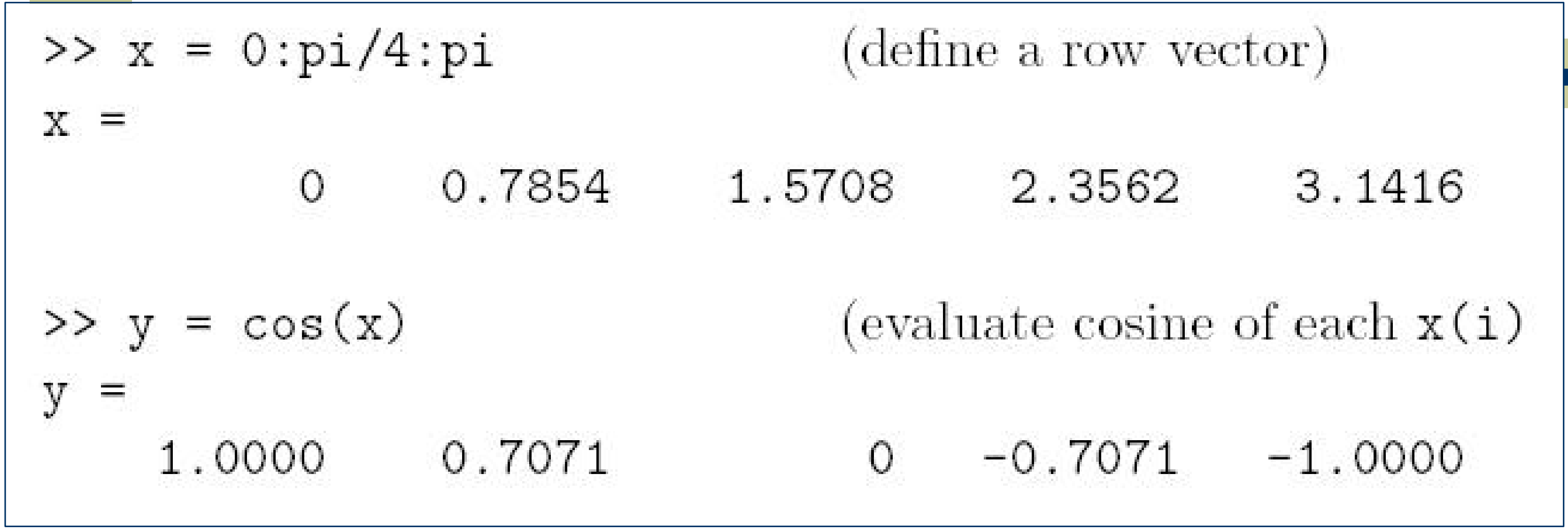


```
>> u = [10 9 8];
>> v = [1 2 3];
>> u+v
ans =
      11      11      11


>> u-v
ans =
       9       7       5
```

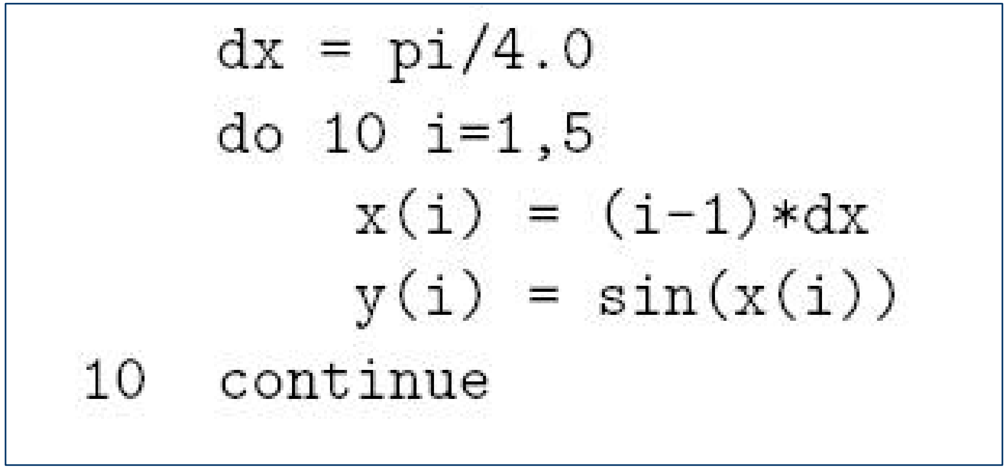


```
>> u = [10 9 8];
>> v = [1 2 3];
>> u*v'
ans =
    52

>> u'*v
ans =
    10    20    30
     9    18    27
     8    16    24
```

# Vectorization

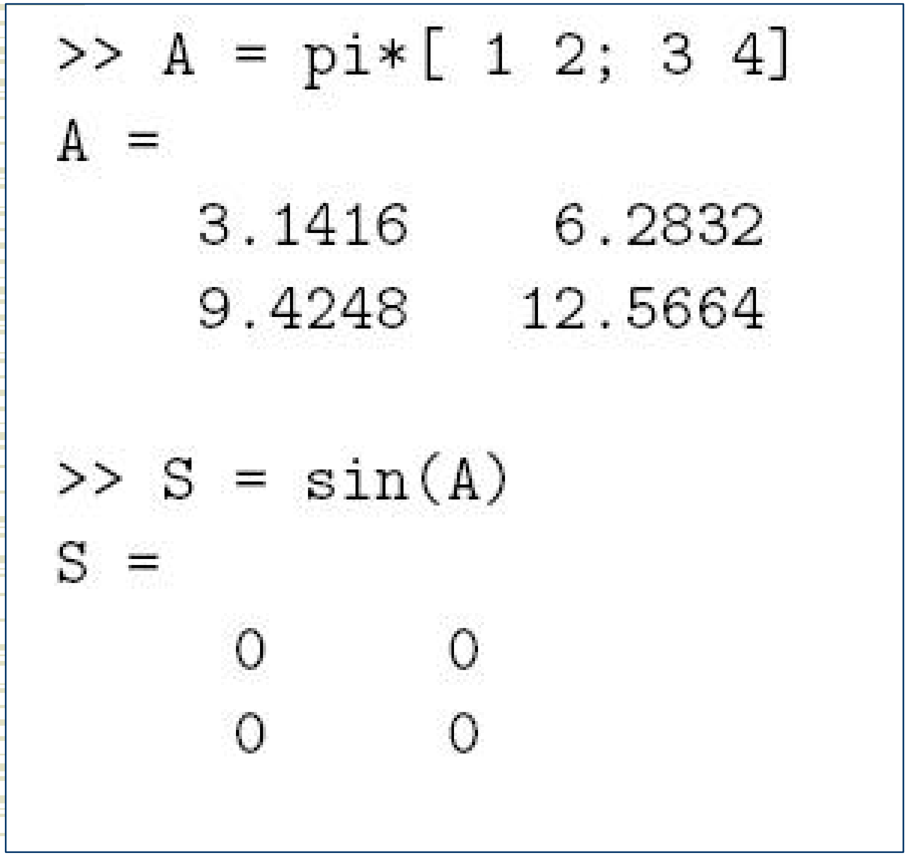


```
>> x = 0:pi/4:pi                (define a row vector)
x =
        0    0.7854    1.5708    2.3562    3.1416

>> y = cos(x)                   (evaluate cosine of each x(i)
y =
   1.0000    0.7071         0   -0.7071   -1.0000
```

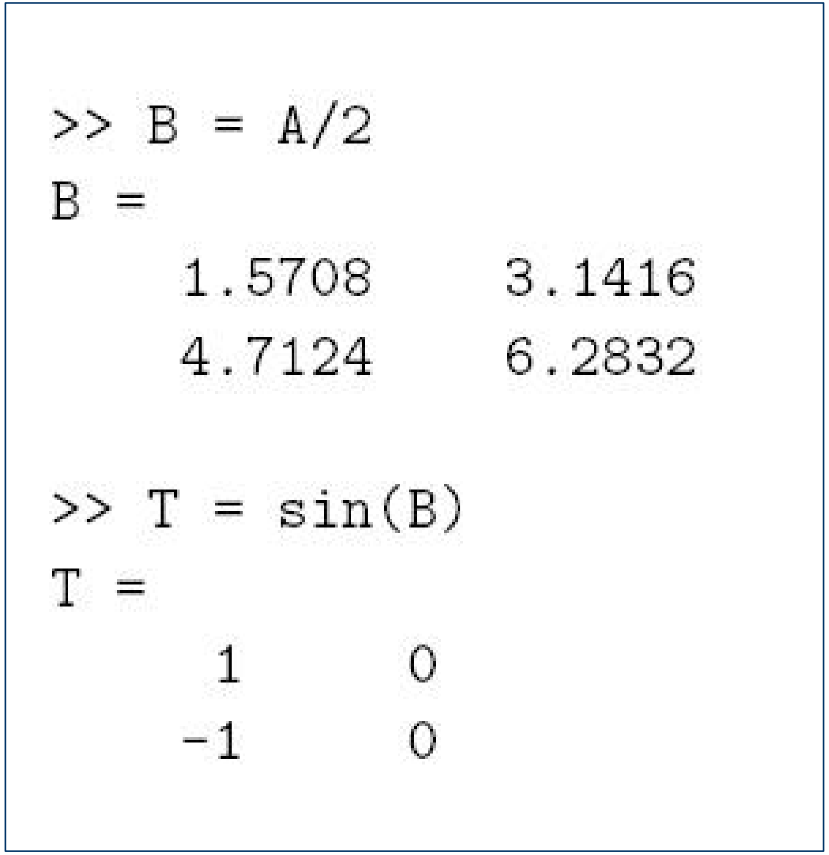


```
      dx = pi/4.0
      do 10 i=1,5
         x(i) = (i-1)*dx
         y(i) = sin(x(i))
 10   continue
```

# Vectorization (Cont.)

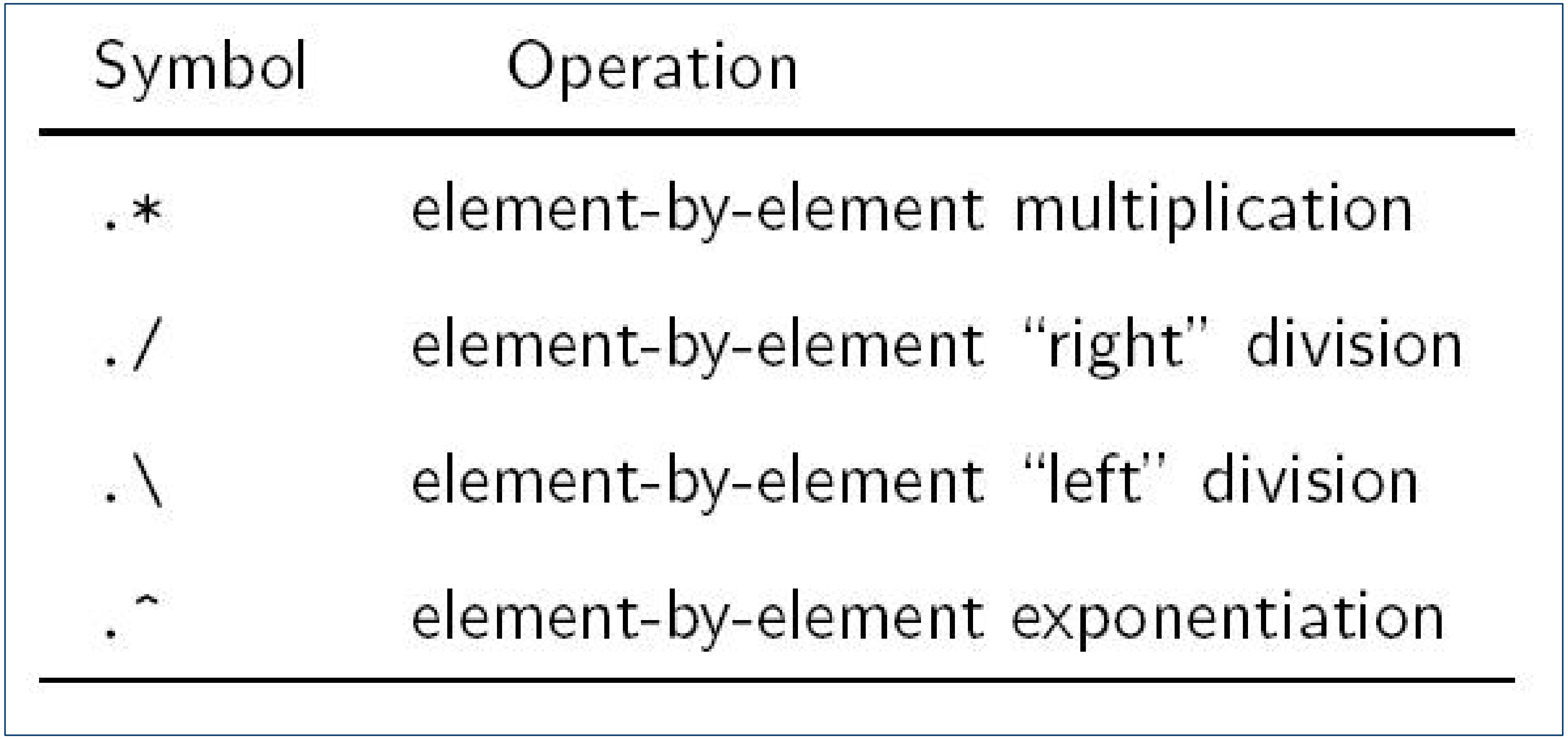


```
>> A = pi*[ 1 2; 3 4]
A =
    3.1416     6.2832
    9.4248    12.5664

>> S = sin(A)
S =
      0         0
      0         0
```



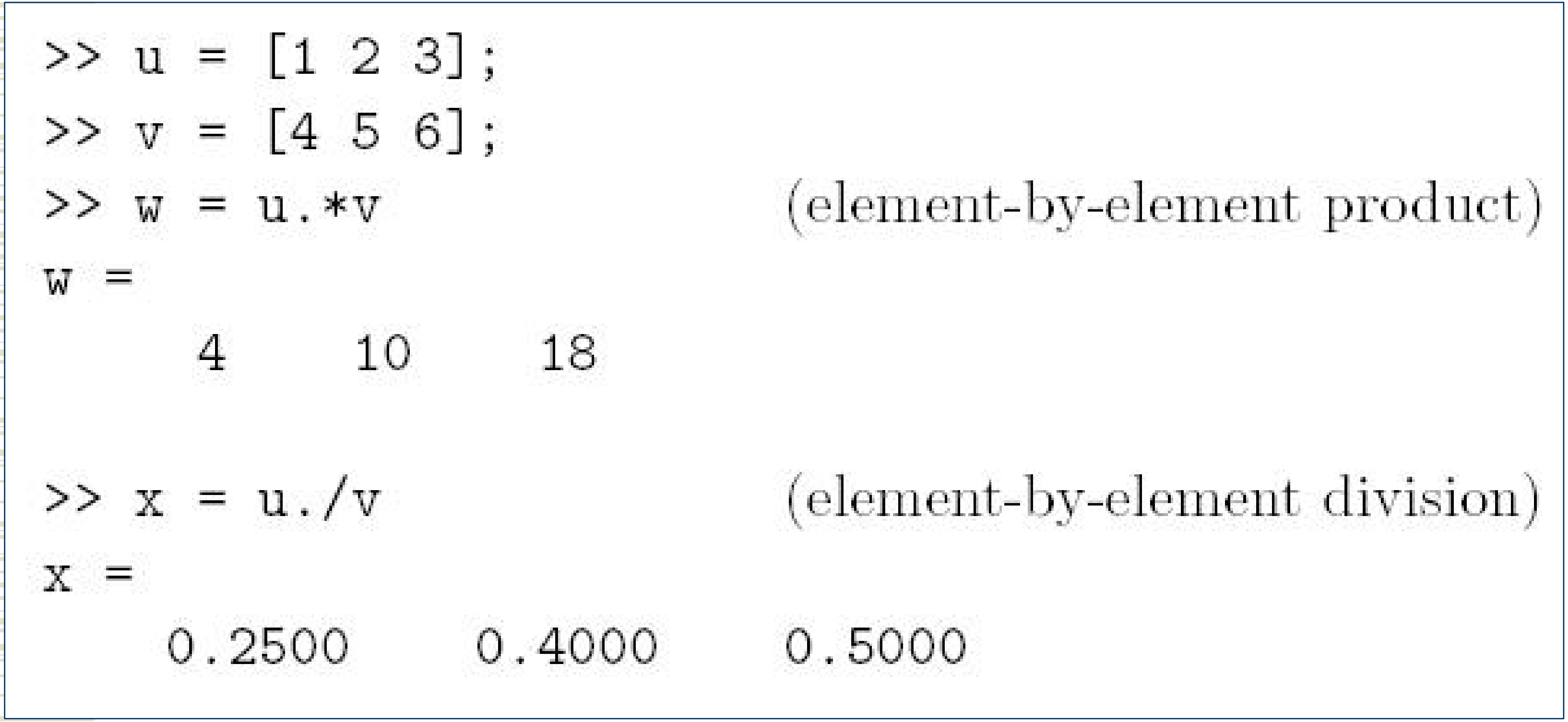
```
>> B = A/2
B =
    1.5708     3.1416
    4.7124     6.2832

>> T = sin(B)
T =
     1        0
    -1        0
```

# Array Operators



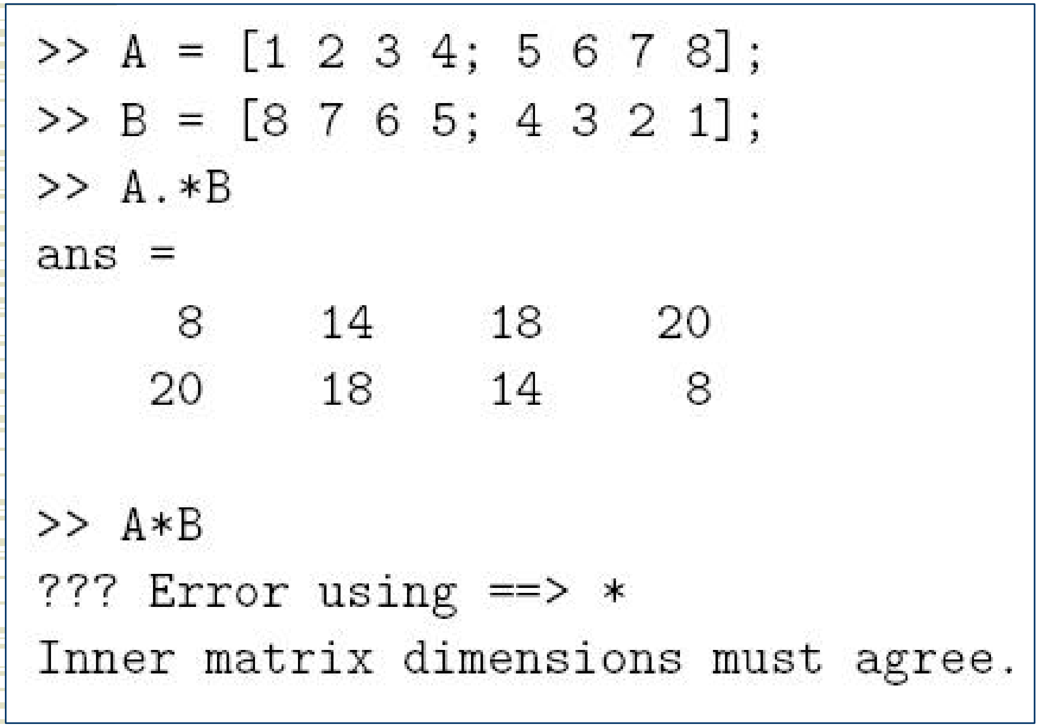
| Symbol | Operation |
|---|---|
| .* | element-by-element multiplication |
| ./ | element-by-element “right” division |
| .\ | element-by-element “left” division |
| .^ | element-by-element exponentiation |

# Array Operators (Cont.)

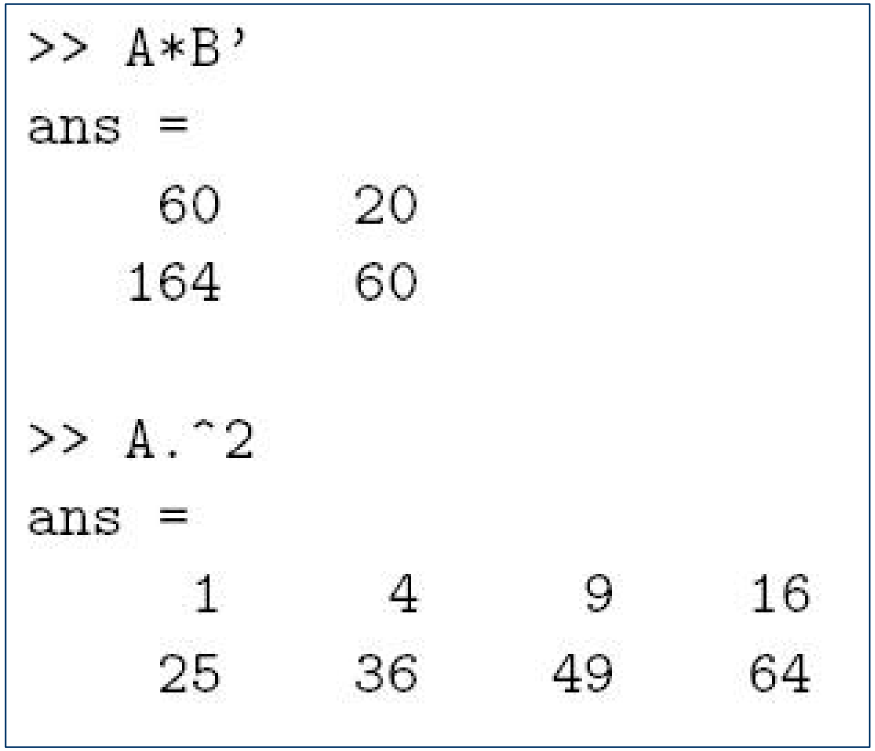


```
>> u = [1 2 3];
>> v = [4 5 6];
>> w = u.*v                    (element-by-element product)
w =
      4     10     18

>> x = u./v                    (element-by-element division)
x =
    0.2500    0.4000    0.5000
```

# Array Operators (Cont.)

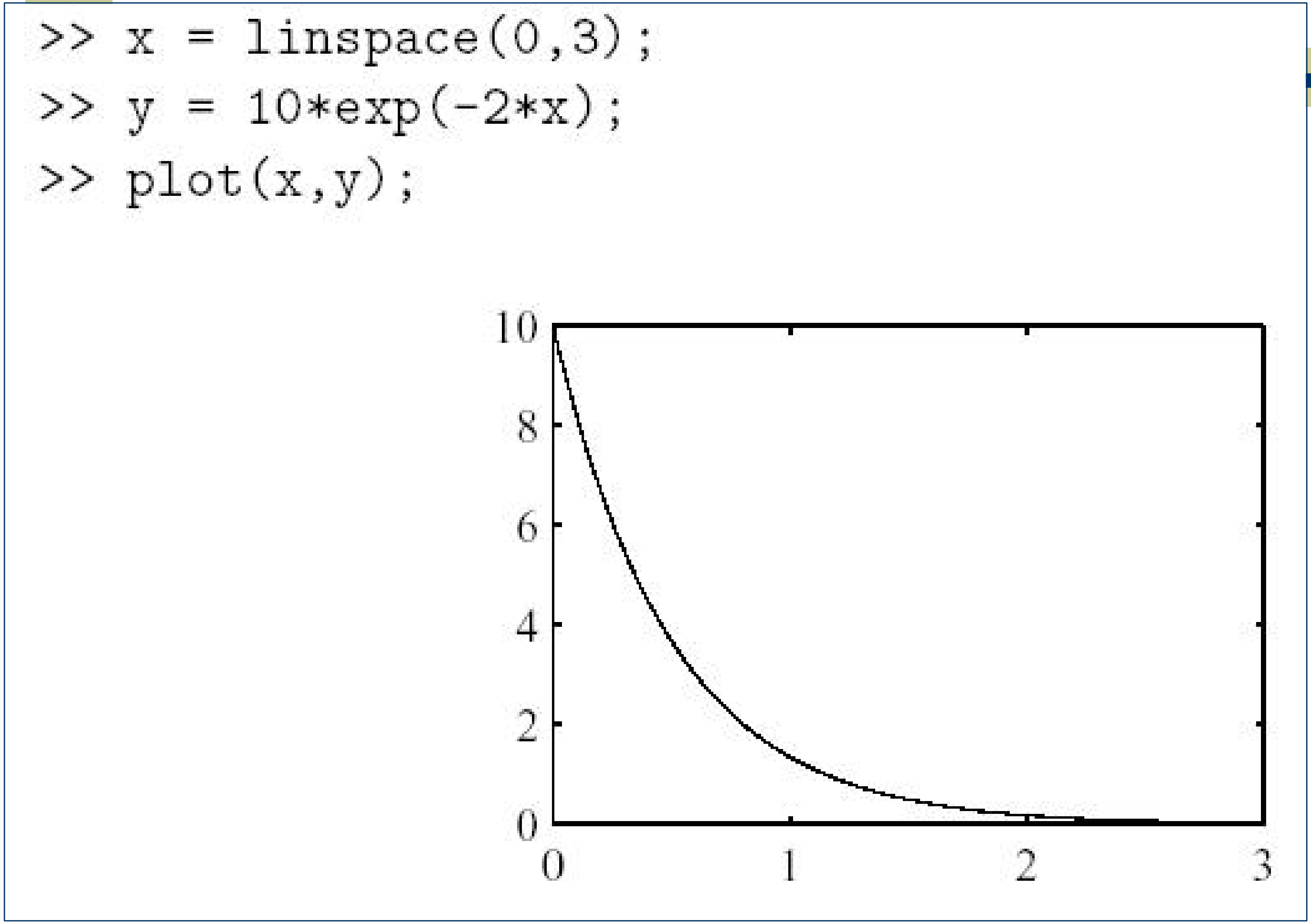


```
>> A = [1 2 3 4; 5 6 7 8];
>> B = [8 7 6 5; 4 3 2 1];
>> A.*B
ans =
     8    14    18    20
    20    18    14     8

>> A*B
??? Error using ==> *
Inner matrix dimensions must agree.
```



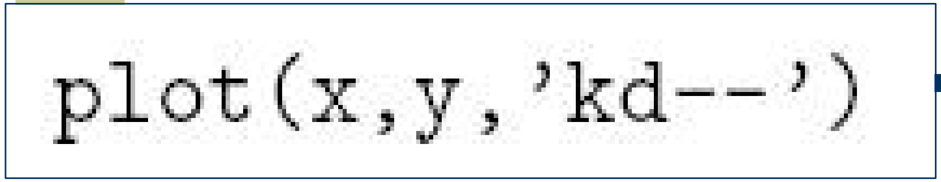
```
>> A*B'
ans =
    60    20
   164    60

>> A.^2
ans =
     1     4     9    16
    25    36    49    64
```

# Plotting

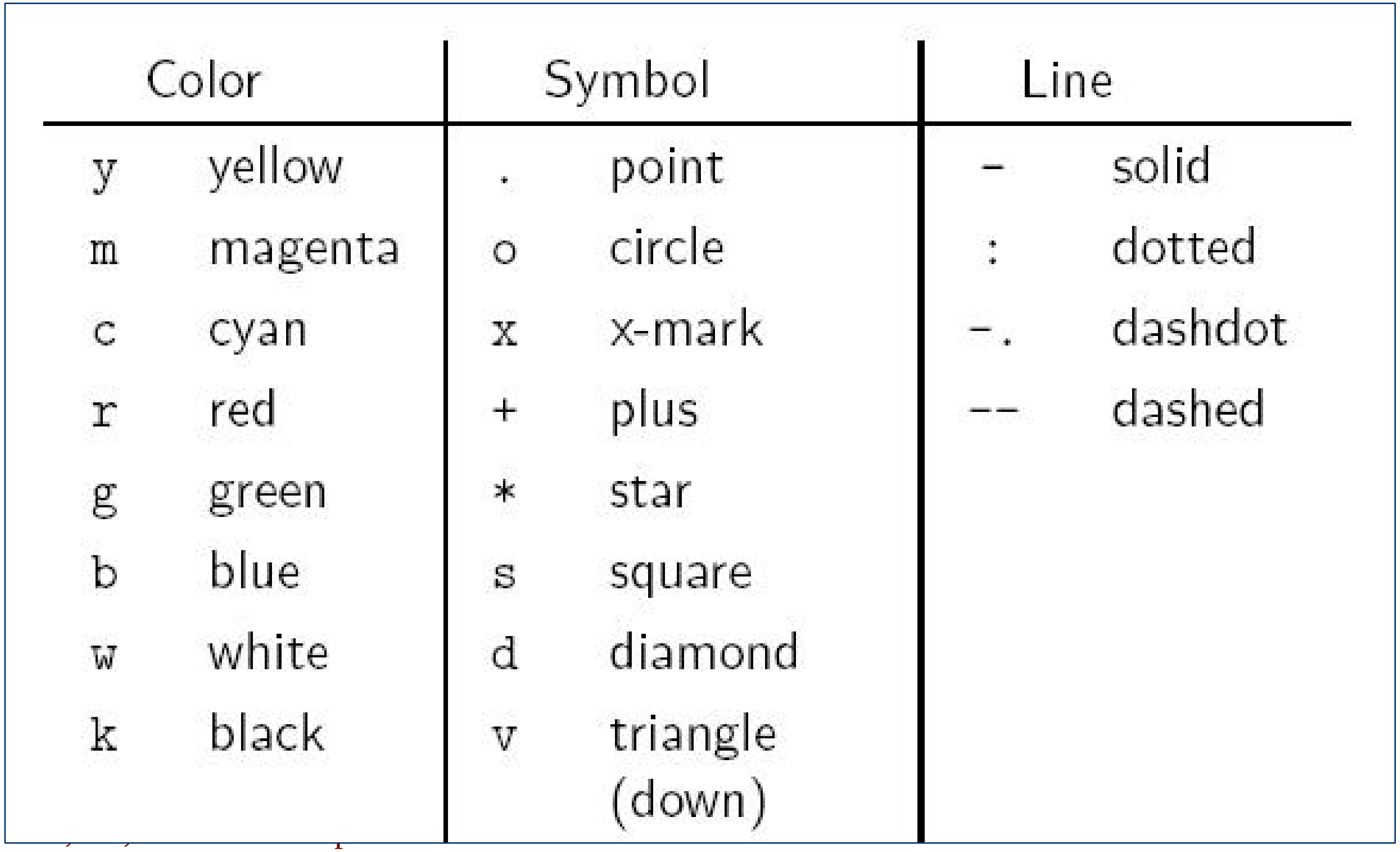


```
>> x = linspace(0,3);
>> y = 10*exp(-2*x);
>> plot(x,y);
```

# Plotting (Cont.)



```
plot(x,y,'kd--')
```



| Color | | Symbol | | Line | |
|---|---|---|---|---|---|
| y | yellow | . | point | - | solid |
| m | magenta | o | circle | : | dotted |
| c | cyan | x | x-mark | -. | dashdot |
| r | red | + | plus | -- | dashed |
| g | green | * | star | | |
| b | blue | s | square | | |
| w | white | d | diamond | | |
| k | black | v | triangle (down) | | |

# Case Study

- Examples by Prof. Hermann Karcher
  - Set 1: FirstConventions.m (script)

- Change current directory to the directory containing the file FirstConventions.m

# Give Orders by Command, Script, and Function

- Script
  - Standard ASCII text files containing a sequence of normal MATLAB commands/statements.
  - The command **`PlotSine`** causes the statements in the file named PlotSine.m to be parsed & executed in order. (Interpreter, not compiler.)
  - % Comments start with "%" characte
  - Variables in a script file are global.

# Give Orders (cont.)

- Function
  - Function files provide extensibility to MATLAB.
  - Usually contains input and output.
  - Variables in a function file are by default local.
  - However, you can declare a variable to be global .

# Flow Control Constructs

- ✍Logic Control: if/ switch

```
if I == J
   A(I,J) = 2;
elseif abs(I-J) == 1
   A(I,J) = -1;
else
   A(I,J) = 0;
end
```

```
switch algorithm
case 'ode23‘
   str = '2nd/3rd order';
case {'ode15s', 'ode23s'}
   str = 'stiff system';
otherwise
   str = 'other algorithm';
end
```

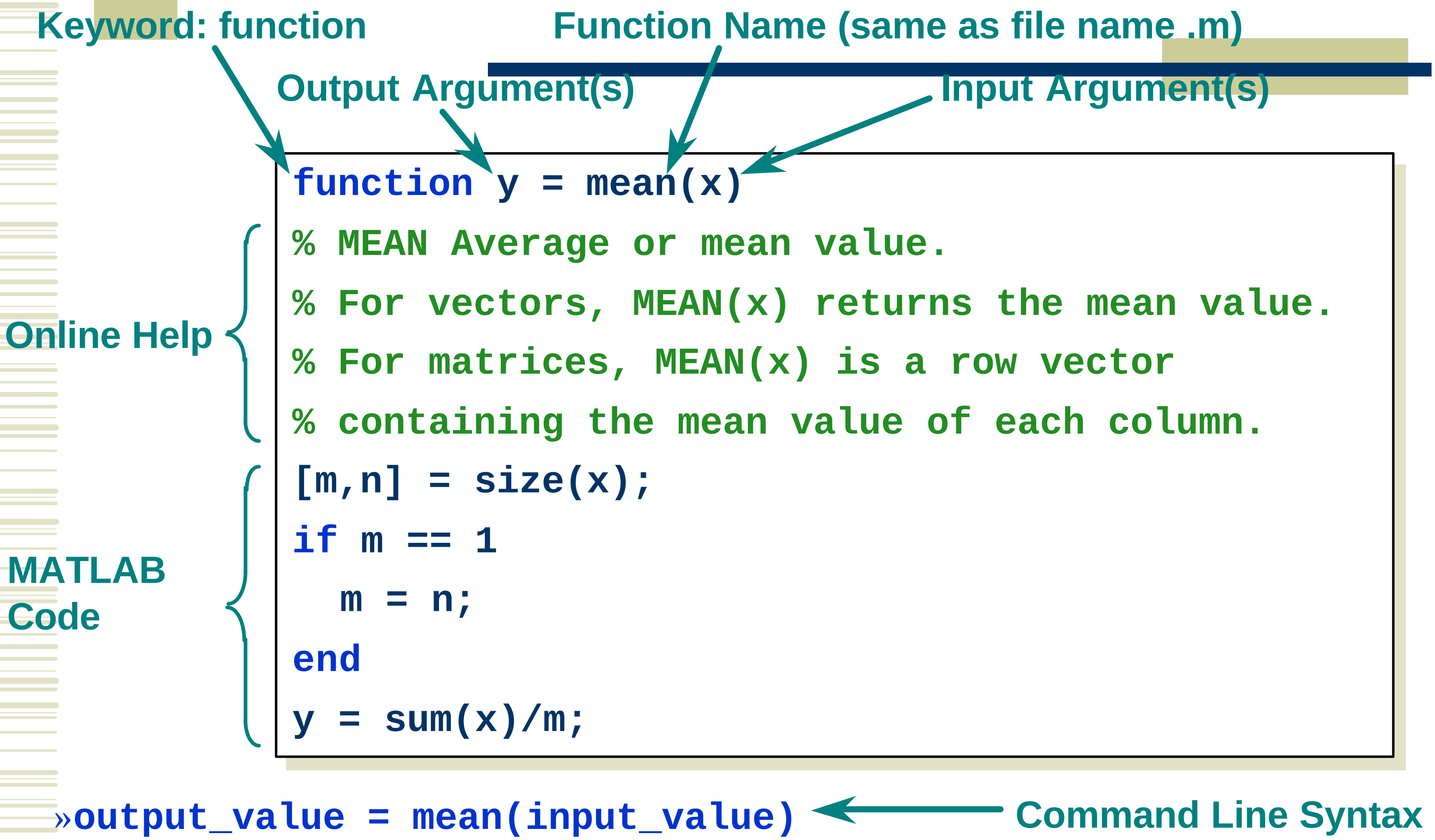
# Flow Control Constructs (cont.)

✍Iterative Loops: for / while

```
N = 10;
for I = 1:N
   for J = 1:N
      A(I,J) = 1/(I+J-1);
   end
end
```

```
I = 1; N = 10;
while I <= N
   J = 1;
   while J <= N
      A(I,J) = 1/(I+J-
      1);
      J = J+1;
   end
   I = I+1;
end
```

# Structure of a Function M-file



Keyword: function

Function Name (same as file name .m)

Output Argument(s)

Input Argument(s)

Online Help

MATLAB Code

```
function y = mean(x)
% MEAN Average or mean value.
% For vectors, MEAN(x) returns the mean value.
% For matrices, MEAN(x) is a row vector
% containing the mean value of each column.
[m,n] = size(x);
if m == 1
  m = n;
end
y = sum(x)/m;
```

```
»output_value = mean(input_value)
```

Command Line Syntax

# Workspaces or Stacks in MATLAB

- ✍MATLAB (or Base) Workspace:

  For command line and script file variables.

- ✍Function Workspaces:

  Each function has its own workspace for local variables.

  Communicate to Function Workspace via inputs & outputs.

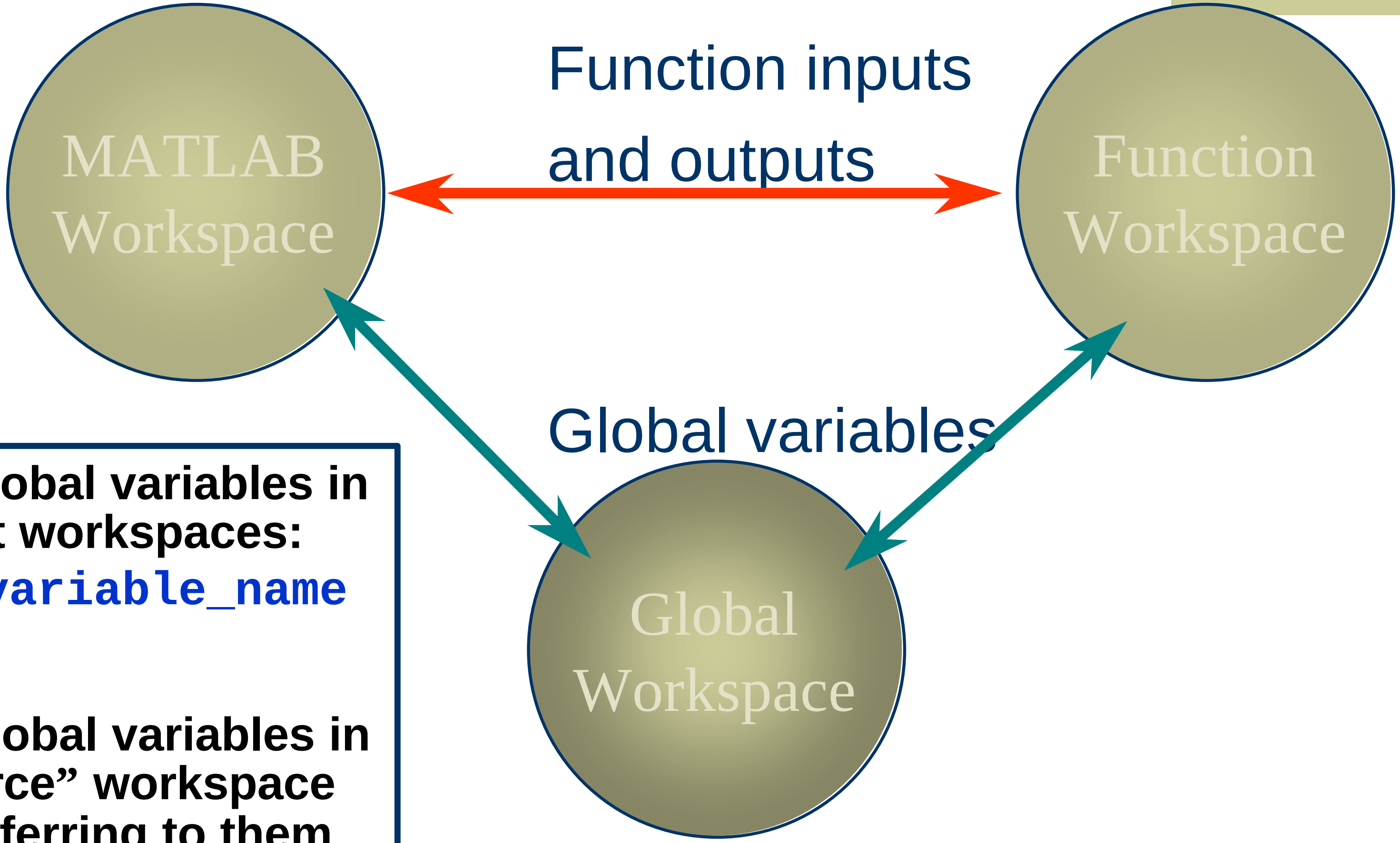
  (*Promotes structured coding & prevents name conflicts*.)

- ✍Global Workspace:

  Global variables can be shared by multiple workspaces.

  (*Must be initialized in all relevant workspaces*.)

# Inter-workspace Communication



Function inputs and outputs

Global variables

**Initialize global variables in all relevant workspaces:**

```
»global variable_name
```

**Initialize global variables in the “source” workspace before referring to them from other workspaces.**

# Case Studies

- ✍Examples by Prof. Hermann Karcher
  - ✍ Set 2: ConCsqrt.m  and Csqrt.m (both functions)

# Some Suggestions

- Write comments

```
% s=1 if scalar product(w,last) >=0  else s=-1
s=2*(real(w.*conj(last))>=0)-1;
```

# Some Suggestions (Cont.)

- ✍Give meaningful variable names and avoid single character variable names

```
a=b*c;
f=m*a;

force = mass * acceleration;
for ii=1:3
  moment = ii^2;
end;
```

# Some Suggestions (Cont.)

- ✍Indent codes
- ✍Use Emacs
  (http://www.gnu.org/software/emacs/emacs.html)
- ✍Learn by doing!

# If You Were Sleeping…

✍It's OK. Just remember two commands:

```
>> lookfor keyword

>> help command
```

Happy coding!

:-)

# Reference

- ✍**鈦思科技股份有限公司 Matlab 簡介投影片**

- ✍Slides by Gerald Recktenwald http://www.prenhall.com/recktenwald.