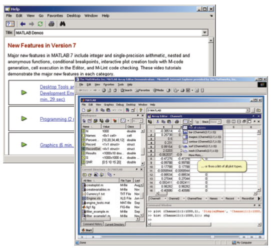
Matlab Programming: Review

Kisung Lee, Ph.D

Medical Information Processing Lab.

KOREA UNIVERSITY

Websites

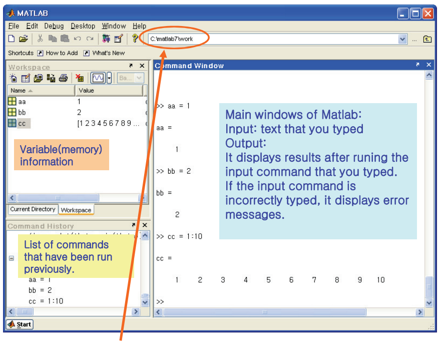


- MATLAB product page
  - http://www.mathworks.com
  - Video tutorial demos
- Demos in MATLAB
  - Video Tutorials
  - Other demos
- Free Webinars
  - http://www.mathworks.com/company/events/webinars/
- Forum, user-made source codes
  - http://www.mathworks.com/matlabcentral/
- Manuals
  - Learning Matlab version 7 (release 14), Mathworks, pdf file, 2005 www.mathworks.com/academia/student_version/learnmatlab_sp3.pdf
  - http://www.mathworks.co.kr/help/techdoc/index.html

Contents

1. Matlab Basics
   - operators, array manipulation
   - plotting
2. Matlab Control Flow
   - conditional statement: if, if-else
   - loops: for
   - character strings
3. Matlab for Programming
   - functions and m files
   - FILE IO
   - simple image processing

Layout of Matlab Windows



Name	Value
aa	1
bb	2
cc	[1 2 3 4 5 6 7 8 9 ...

Variable(memory) information

List of commands that have been run previously.

```
aa = 1
bb = 2
cc = 1:10
```

```
>> aa = 1

aa =

     1

>> bb = 2

bb =

     2

>> cc = 1:10

cc =

     1     2     3     4     5     6     7     8     9    10

>>
```

Main windows of Matlab:
Input: text that you typed
Output:
It displays results after runing the input command that you typed.
If the input command is incorrectly typed, it displays error messages.

- The first thing you MUST do when this window pops up on the screen, is to change the work folder from default(c:\matlab7\work) to your data folder.

Matlab vs. C/C++

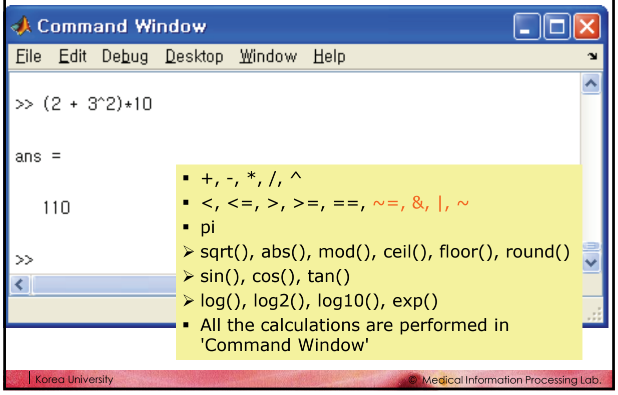
- much easier to program than C/C++
- no need to compile & link: interpreter
- easy to debug
- easy to plot in 2D and 3D
- enormous amount of functions for various applications
- **VERY SLOW when you use loops**

  → not appropriate for applications that require massive calculations or many repeatations such as Monte-Carlo simulation and so on.

Command Window

- System commands(=commands from Win7, Linux, UNIX, etc) that you can run in the 'Command Window'.
  - cd, ls, dir, mkdir, pwd, exit, quit
  - whos, who *% list variables*
  - clear, close
  - help
  - which *% returns the absolute path of functions or files*
  - what *% list matlab files in current directory*
  - lookfor *% search the keyword in help messages in M files*
  - date, now *% current time*

Matlab as a Calculator



```
>> (2 + 3^2)*10

ans =

   110

>>
```

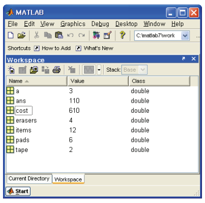
- +, -, *, /, ^
- <, <=, >, >=, ==, ~=, &, |, ~
- pi
- ➢ sqrt(), abs(), mod(), ceil(), floor(), round()
- ➢ sin(), cos(), tan()
- ➢ log(), log2(), log10(), exp()
- All the calculations are performed in 'Command Window'

How to Deal with 1D Array

- [], (), :, end
- ', linspace(), 1:2:10
- array index : starts from 1 (0 in C/C++, IDL)

```
v = [1 3 5 7 9];  v = 1:2:9;
v(2)
w = v'
v(1:3)
v(3:end), v(1:2:end), v(end:-2:1)
a=0; b=10; n=5;
x = linspace(a,b,n) % a~b, # of steps: n
y = [v x]
z = [v, x]
```

Matlab Variables


MATLAB
File Edit View Graphics Debug Desktop Window Help
C:\matlab7\work
Shortcuts How to Add What's New
Workspace
Stack: Base
Current Directory
Workspace
Start

Name	Value	Class
a	3	double
ans	110	double
cost	610	double
erasers	4	double
items	12	double
pads	6	double
tape	2	double

- =
- ;
  - ➢ a = 3
  - ➢ a = 3;
  - ➢ a
- 33.8/7 + 9.5*2.9 = ?
  - ➢ t1 = 33.8 / 7; t2 = 9.6 * 2.9; total = t1+t2

  How can we compute the answer with a single command? → Storing 3 lines into a file (e.g. comp.m) and type the filename at the command prompt. (See next page.)
- Example
  - ➢ erasers=4; pads=6; tape=2;
  - ➢ items=erasers + pads + tape
  - ➢ cost=erasers*25 + pads*52 + tape*99
- All the variables allocated by you are listed in 'Workspace' window.

How to Deal with 2D Array

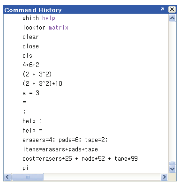
$$f = \begin{pmatrix} f(1,1) & f(1,2) & \dots & f(1,N) \\ f(2,1) & f(2,2) & \dots & f(2,N) \\ \vdots & & & \\ f(M,1) & f(M,2) & \dots & f(M,N) \end{pmatrix}$$

- MxN (row x col) matrix
- Mx1 matrix : column vector
- 1xN matrix : row vector
- 1x1 matrix : scalar

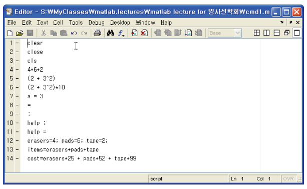
- ; again
- Matrix dimension : # rows x # columns
  - ➢ a = [1 2 3; 4 5 6; 7 8 9]
  - ➢ a(2,3)
  - ➢ a(:,3), a(2,:), a(1:2, 1:3)
  - ➢ a(end,end), a(end, end-2), a(1:2:end, end:-2:1)
  - ➢ a(:) = 0; a(:,:) = 0;

M Files for Batch Job

- All the commands you have typed are listed in 'Command History' window.
- The hostory can be edited and stored in a file with extension '.m'.
- In order to run the commands in batch, type the M-file name.
- Make sure that the path is valid.
- % in M-file text



```
which help
lookfor matrix
clear
close
cls
4+6+2
(2 + 3^2)
(2 + 3^2)*10
a = 3
=
;
help ;
help =
erasers=4; pads=6; tape=2;
items=erasers+pads+tape
cost=erasers*25 + pads*52 + tape*99
pi
```

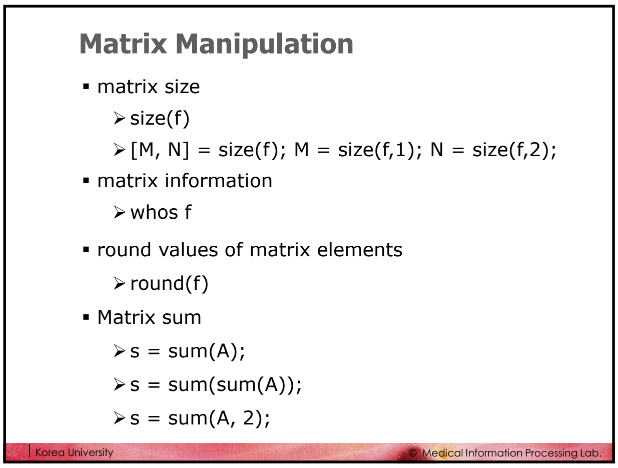

Editor - S:₩MyClasses₩matlab.lectures₩matlab lecture for 방사선학회₩cmd1.m
File Edit Text Cell Tools Debug Desktop Window Help
Base
script
Ln 1
Col 1
OVR

```
clear
close
cls
4+6+2
(2 + 3^2)
(2 + 3^2)*10
a = 3
=
;
help ;
help =
erasers=4; pads=6; tape=2;
items=erasers+pads+tape
cost=erasers*25 + pads*52 + tape*99
```

Matrix Generation and Operation

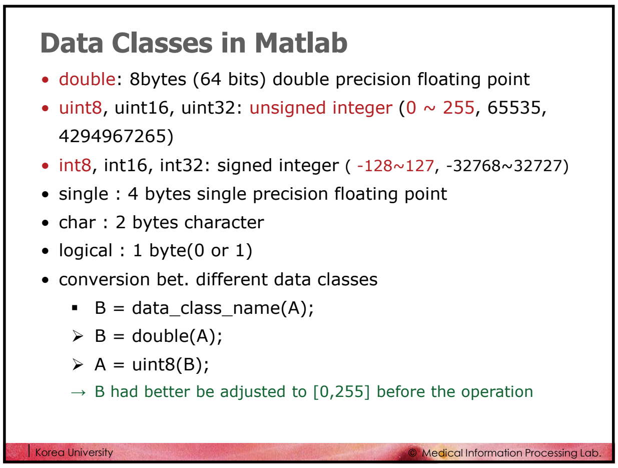
- ➢ zeros(M,N) : double
- ➢ ones(M,N) : double
- ➢ true(M,N) : logical, all elements are 1
- ➢ false(M,N) : logical, all elements are 0
- ➢ rand(M,N) : random numbers [0,1]
- examples: A = 5 * ones(3,3); B = rand(2,4)

- point operators for matrix : .*, ./, .^
- *, /, ^ : difference ?



Matrix Manipulation

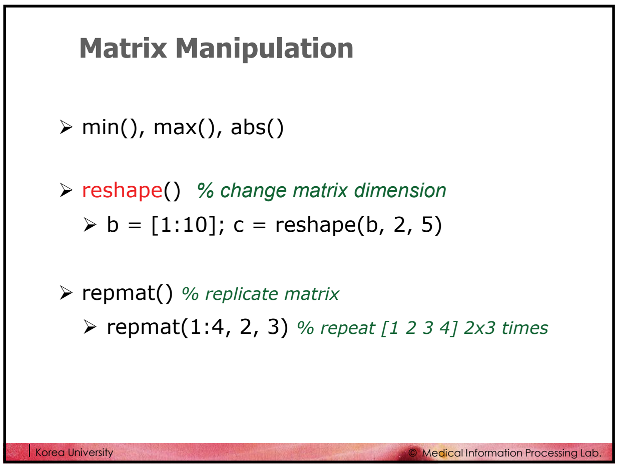
- matrix size
  - size(f)
  - [M, N] = size(f); M = size(f,1); N = size(f,2);
- matrix information
  - whos f
- round values of matrix elements
  - round(f)
- Matrix sum
  - s = sum(A);
  - s = sum(sum(A));
  - s = sum(A, 2);



Data Classes in Matlab

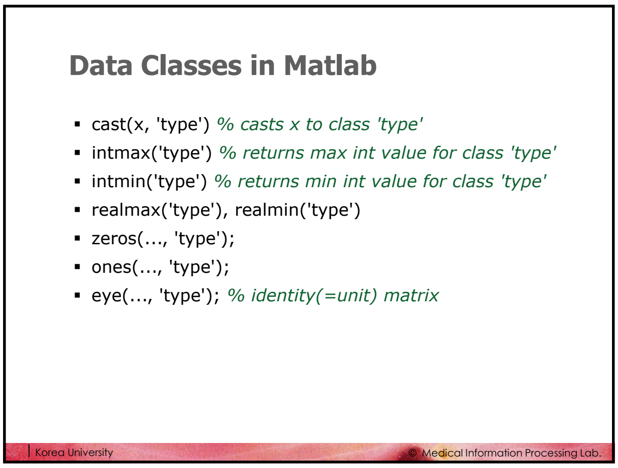
- double: 8bytes (64 bits) double precision floating point
- uint8, uint16, uint32: unsigned integer (0 ~ 255, 65535, 4294967265)
- int8, int16, int32: signed integer (-128~127, -32768~32727)
- single : 4 bytes single precision floating point
- char : 2 bytes character
- logical : 1 byte(0 or 1)
- conversion bet. different data classes
  - B = data_class_name(A);
  - B = double(A);
  - A = uint8(B);

  → B had better be adjusted to [0,255] before the operation



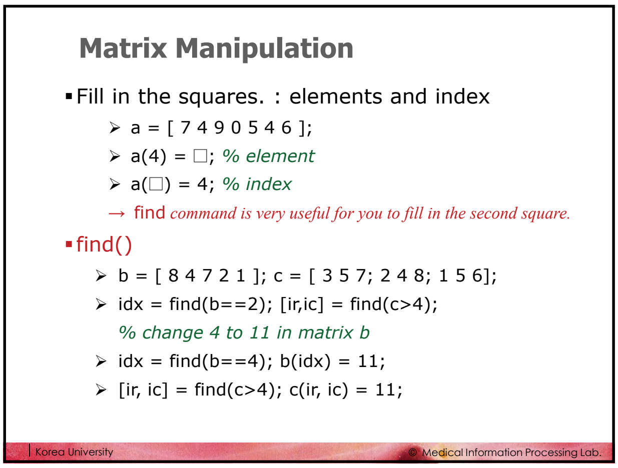
Matrix Manipulation

- min(), max(), abs()
- reshape() *% change matrix dimension*
  - b = [1:10]; c = reshape(b, 2, 5)
- repmat() *% replicate matrix*
  - repmat(1:4, 2, 3) *% repeat [1 2 3 4] 2x3 times*



Data Classes in Matlab

- cast(x, 'type') *% casts x to class 'type'*
- intmax('type') *% returns max int value for class 'type'*
- intmin('type') *% returns min int value for class 'type'*
- realmax('type'), realmin('type')
- zeros(…, 'type');
- ones(…, 'type');
- eye(…, 'type'); *% identity(=unit) matrix*

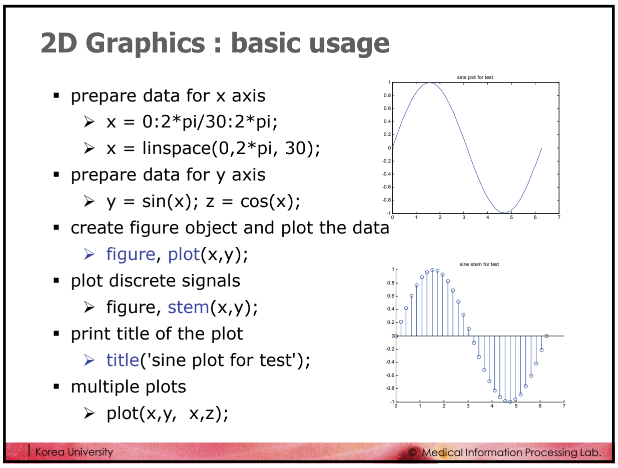


Matrix Manipulation

- Fill in the squares. : elements and index
  - a = [7 4 9 0 5 4 6];
  - a(4) = □; *% element*
  - a(□) = 4; *% index*

  → find *command is very useful for you to fill in the second square.*
- find()
  - b = [8 4 7 2 1]; c = [3 5 7; 2 4 8; 1 5 6];
  - idx = find(b==2); [ir,ic] = find(c>4);

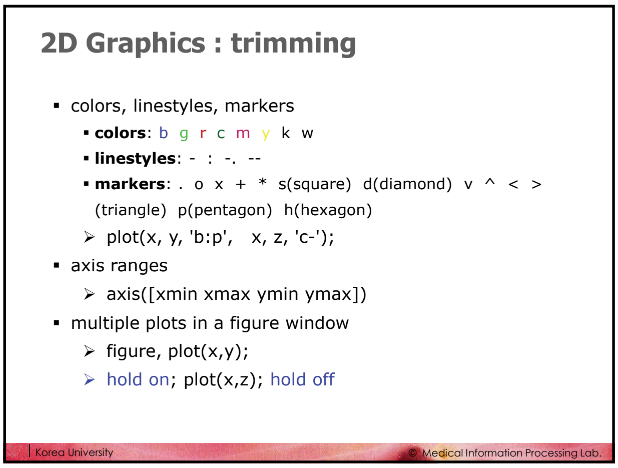
    % change 4 to 11 in matrix b
  - idx = find(b==4); b(idx) = 11;
  - [ir, ic] = find(c>4); c(ir, ic) = 11;



2D Graphics : basic usage

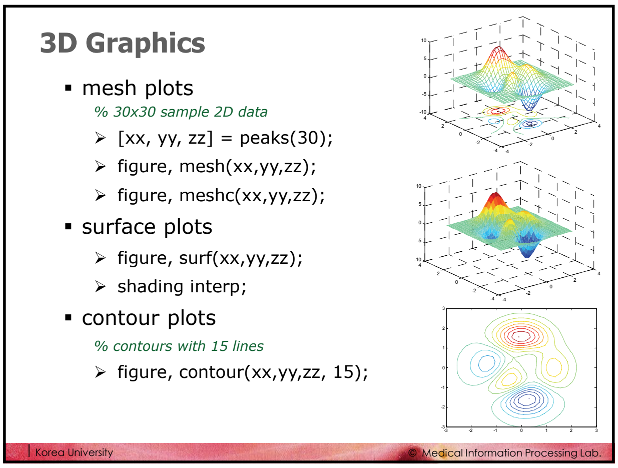
- prepare data for x axis
  - x = 0:2*pi/30:2*pi;
  - x = linspace(0,2*pi, 30);
- prepare data for y axis
  - y = sin(x); z = cos(x);
- create figure object and plot the data
  - figure, plot(x,y);
- plot discrete signals
  - figure, stem(x,y);
- print title of the plot
  - title('sine plot for test');
- multiple plots
  - plot(x,y, x,z);

sine plot for test
1
0.8
0.6
0.4
0.2
0
-0.2
-0.4
-0.6
-0.8
-1
0
1
2
3
4
5
6
7
sine stem for test
1
0.8
0.6
0.4
0.2
0
-0.2
-0.4
-0.6
-0.8
-1
0
1
2
3
4
5
6
7



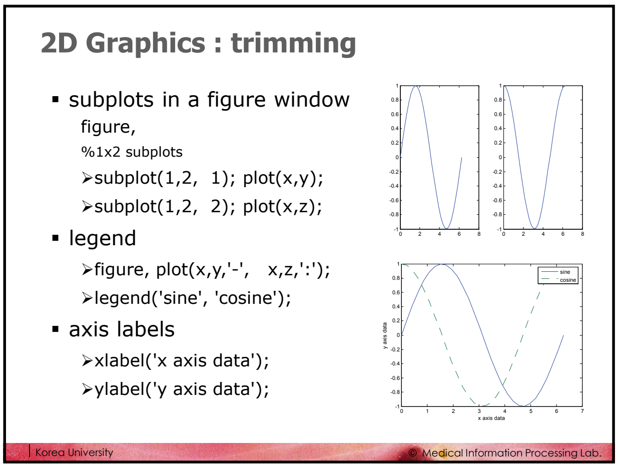
2D Graphics : trimming

- colors, linestyles, markers
  - **colors**: b g r c m y k w
  - **linestyles**: - : -. --
  - **markers**: . o x + * s(square) d(diamond) v ^ < > (triangle) p(pentagon) h(hexagon)
  - ➢ plot(x, y, 'b:p', x, z, 'c-');
- axis ranges
  - ➢ axis([xmin xmax ymin ymax])
- multiple plots in a figure window
  - ➢ figure, plot(x,y);
  - ➢ hold on; plot(x,z); hold off



3D Graphics

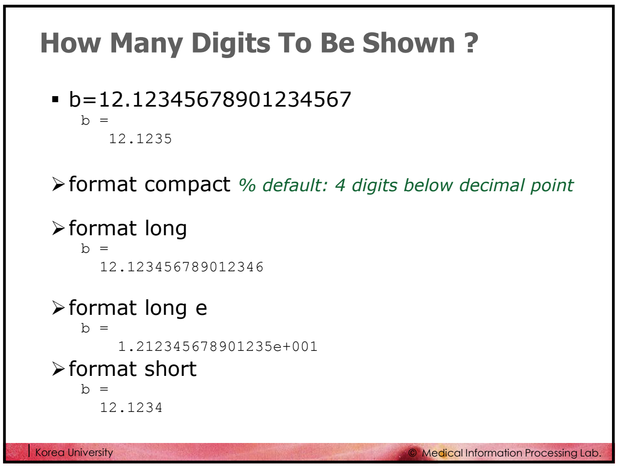
- mesh plots
  - *% 30x30 sample 2D data*
  - ➢ [xx, yy, zz] = peaks(30);
  - ➢ figure, mesh(xx,yy,zz);
  - ➢ figure, meshc(xx,yy,zz);
- surface plots
  - ➢ figure, surf(xx,yy,zz);
  - ➢ shading interp;
- contour plots
  - *% contours with 15 lines*
  - ➢ figure, contour(xx,yy,zz, 15);



2D Graphics : trimming

- subplots in a figure window
  - figure,
  - %1x2 subplots
  - ➢subplot(1,2, 1); plot(x,y);
  - ➢subplot(1,2, 2); plot(x,z);
- legend
  - ➢figure, plot(x,y,'-', x,z,':');
  - ➢legend('sine', 'cosine');
- axis labels
  - ➢xlabel('x axis data');
  - ➢ylabel('y axis data');

sine
cosine
x axis data
y axis data



How Many Digits To Be Shown ?

- b=12.12345678901234567

```
b =
    12.1235
```

➢format compact *% default: 4 digits below decimal point*

➢format long

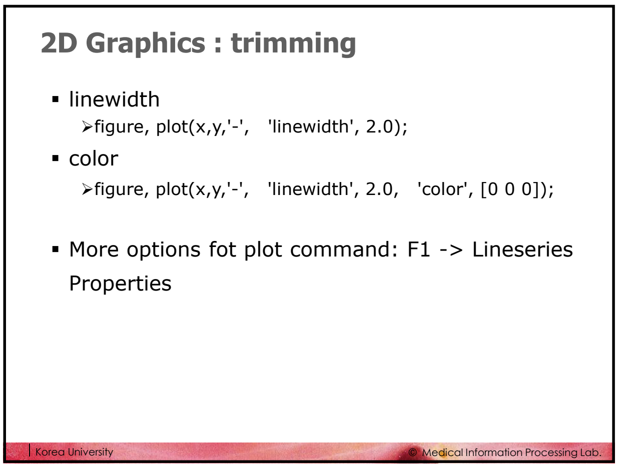
```
b =
   12.123456789012346
```

➢format long e

```
b =
     1.212345678901235e+001
```

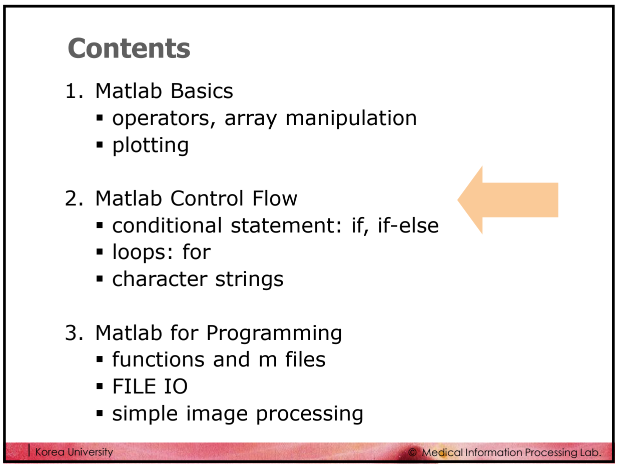
➢format short

```
b =
   12.1234
```



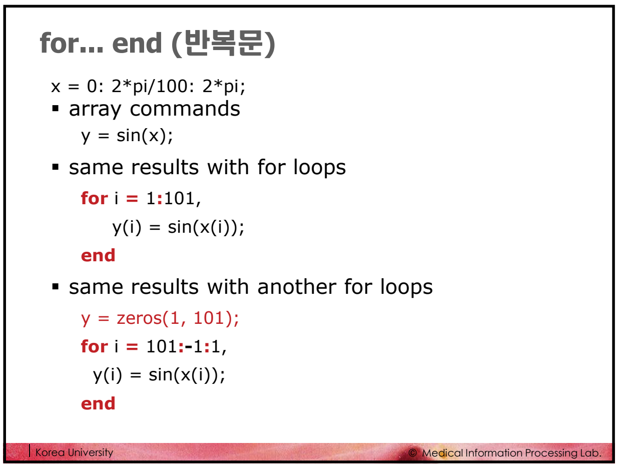
2D Graphics : trimming

- linewidth
  - ➢figure, plot(x,y,'-', 'linewidth', 2.0);
- color
  - ➢figure, plot(x,y,'-', 'linewidth', 2.0, 'color', [0 0 0]);
- More options fot plot command: F1 -> Lineseries Properties



Contents

1. Matlab Basics
   - operators, array manipulation
   - plotting
2. Matlab Control Flow
   - conditional statement: if, if-else
   - loops: for
   - character strings
3. Matlab for Programming
   - functions and m files
   - FILE IO
   - simple image processing



for... end (반복문)

x = 0: 2*pi/100: 2*pi;

- array commands

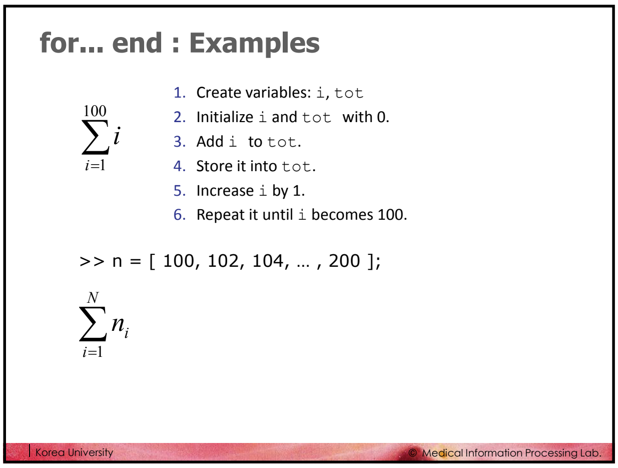
```
y = sin(x);
```

- same results with for loops

```
for i = 1:101,
    y(i) = sin(x(i));
end
```

- same results with another for loops

```
y = zeros(1, 101);
for i = 101:-1:1,
  y(i) = sin(x(i));
end
```



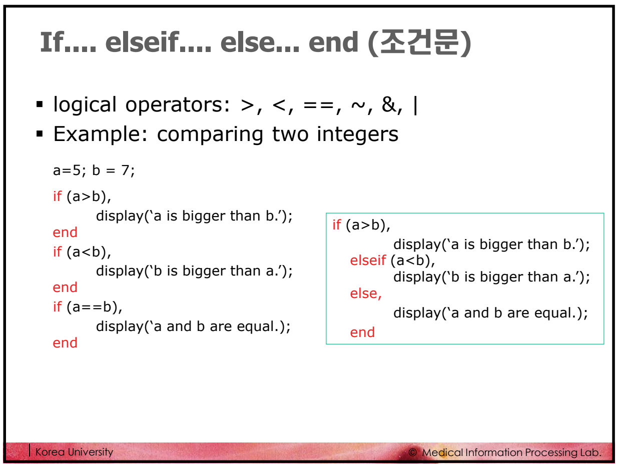
for... end : Examples

$$\sum_{i=1}^{100} i$$

1. Create variables: `i`, `tot`
2. Initialize `i` and `tot` with 0.
3. Add `i` to `tot`.
4. Store it into `tot`.
5. Increase `i` by 1.
6. Repeat it until `i` becomes 100.

```
>> n = [ 100, 102, 104, … , 200 ];
```

$$\sum_{i=1}^{N} n_i$$

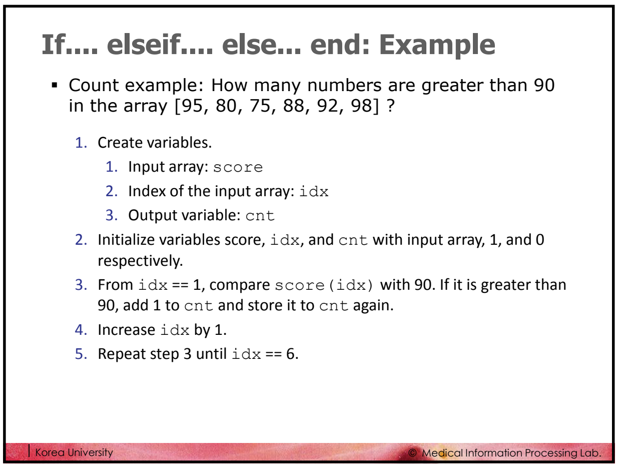


If…. elseif…. else… end (조건문)

- logical operators: >, <, ==, ~, &, |
- Example: comparing two integers

```
a=5; b = 7;
if (a>b),
        display(‘a is bigger than b.’);
end
if (a<b),
        display(‘b is bigger than a.’);
end
if (a==b),
        display(‘a and b are equal.);
end
```

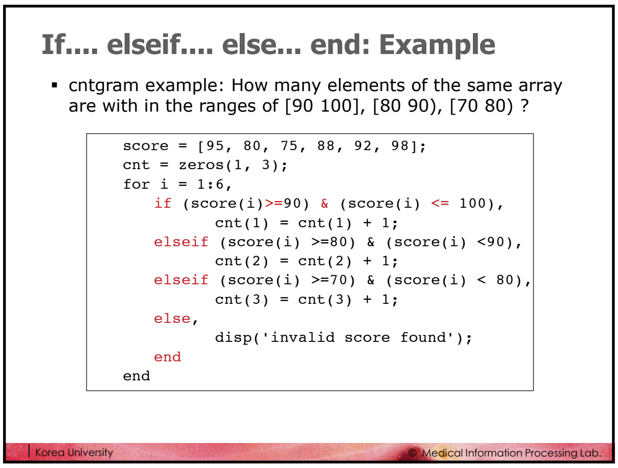
```
if (a>b),
        display(‘a is bigger than b.’);
   elseif (a<b),
        display(‘b is bigger than a.’);
   else,
        display(‘a and b are equal.);
   end
```



If…. elseif…. else… end: Example

- Count example: How many numbers are greater than 90 in the array [95, 80, 75, 88, 92, 98] ?

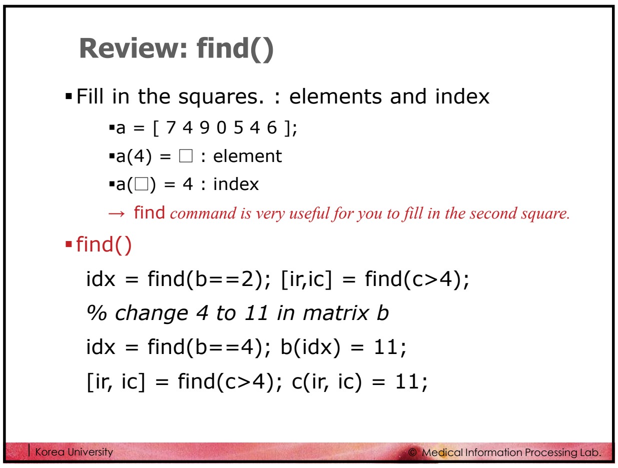
1. Create variables.
   1. Input array: `score`
   2. Index of the input array: `idx`
   3. Output variable: `cnt`
2. Initialize variables score, `idx`, and `cnt` with input array, 1, and 0 respectively.
3. From `idx` == 1, compare `score(idx)` with 90. If it is greater than 90, add 1 to `cnt` and store it to `cnt` again.
4. Increase `idx` by 1.
5. Repeat step 3 until `idx` == 6.



If…. elseif…. else… end: Example

- cntgram example: How many elements of the same array are with in the ranges of [90 100], [80 90), [70 80) ?

```
score = [95, 80, 75, 88, 92, 98];
cnt = zeros(1, 3);
for i = 1:6,
    if (score(i)>=90) & (score(i) <= 100),
            cnt(1) = cnt(1) + 1;
    elseif (score(i) >=80) & (score(i) <90),
            cnt(2) = cnt(2) + 1;
    elseif (score(i) >=70) & (score(i) < 80),
            cnt(3) = cnt(3) + 1;
    else,
            disp('invalid score found');
    end
end
```



Review: find()

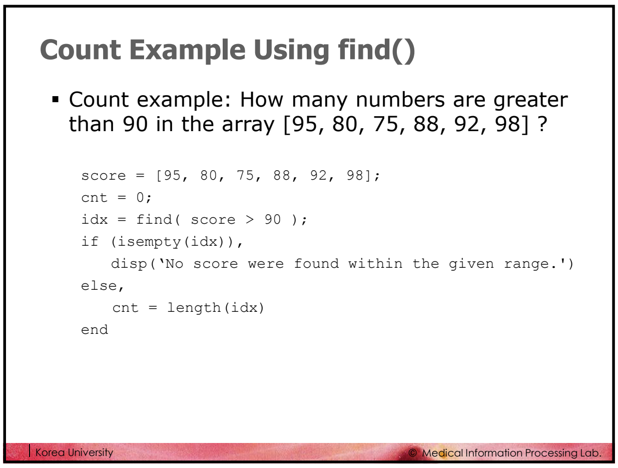
- Fill in the squares. : elements and index
  - a = [7 4 9 0 5 4 6];
  - a(4) = □ : element
  - a(□) = 4 : index
  - → find *command is very useful for you to fill in the second square.*
- find()

idx = find(b==2); [ir,ic] = find(c>4);

% change 4 to 11 in matrix b

idx = find(b==4); b(idx) = 11;

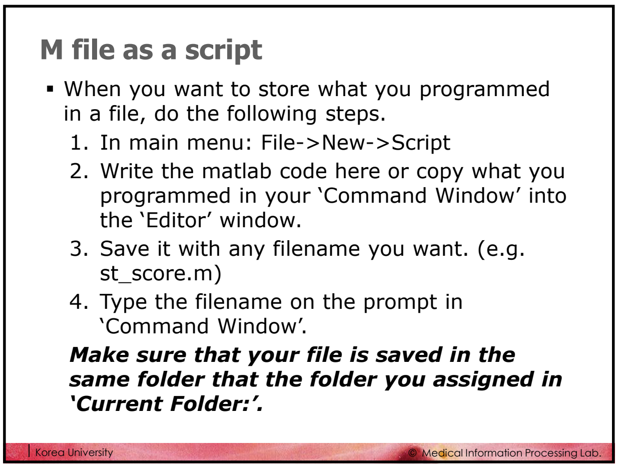
[ir, ic] = find(c>4); c(ir, ic) = 11;



Count Example Using find()

- Count example: How many numbers are greater than 90 in the array [95, 80, 75, 88, 92, 98] ?

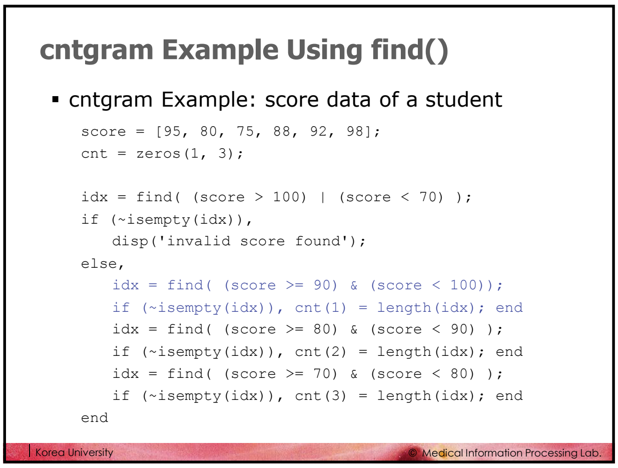
```
score = [95, 80, 75, 88, 92, 98];
cnt = 0;
idx = find( score > 90 );
if (isempty(idx)),
   disp('No score were found within the given range.')
else,
   cnt = length(idx)
end
```



M file as a script

- When you want to store what you programmed in a file, do the following steps.
  1. In main menu: File->New->Script
  2. Write the matlab code here or copy what you programmed in your 'Command Window' into the 'Editor' window.
  3. Save it with any filename you want. (e.g. st_score.m)
  4. Type the filename on the prompt in 'Command Window'.

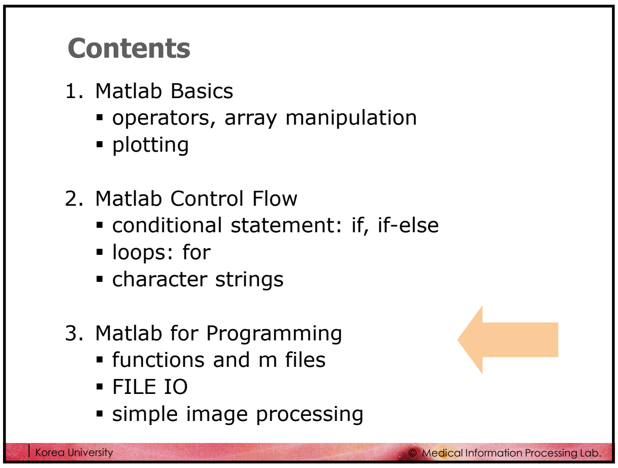
  Make sure that your file is saved in the same folder that the folder you assigned in 'Current Folder:'.



cntgram Example Using find()

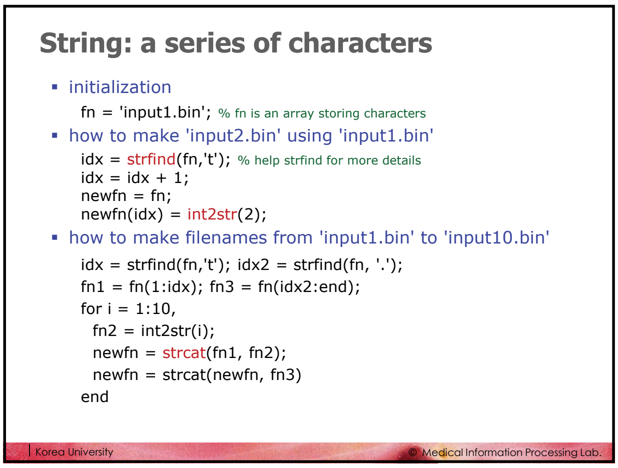
- cntgram Example: score data of a student

```
score = [95, 80, 75, 88, 92, 98];
cnt = zeros(1, 3);

idx = find( (score > 100) | (score < 70) );
if (~isempty(idx)),
   disp('invalid score found');
else,
   idx = find( (score >= 90) & (score < 100));
   if (~isempty(idx)), cnt(1) = length(idx); end
   idx = find( (score >= 80) & (score < 90) );
   if (~isempty(idx)), cnt(2) = length(idx); end
   idx = find( (score >= 70) & (score < 80) );
   if (~isempty(idx)), cnt(3) = length(idx); end
end
```



Contents

1. Matlab Basics
   - operators, array manipulation
   - plotting
2. Matlab Control Flow
   - conditional statement: if, if-else
   - loops: for
   - character strings
3. Matlab for Programming
   - functions and m files
   - FILE IO
   - simple image processing



String: a series of characters

- initialization

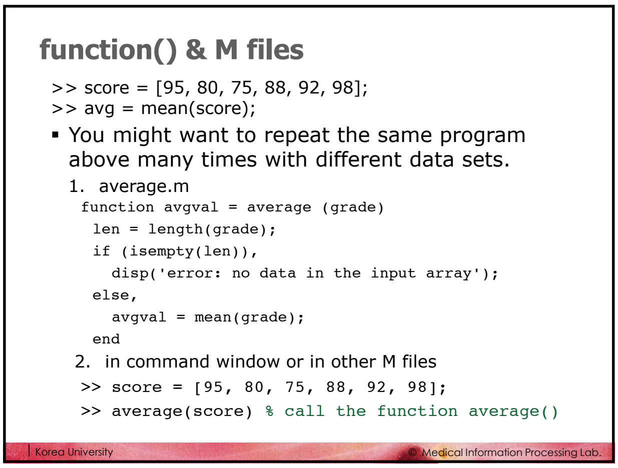
```
fn = 'input1.bin'; % fn is an array storing characters
```

- how to make 'input2.bin' using 'input1.bin'

```
idx = strfind(fn,'t'); % help strfind for more details
idx = idx + 1;
newfn = fn;
newfn(idx) = int2str(2);
```

- how to make filenames from 'input1.bin' to 'input10.bin'

```
idx = strfind(fn,'t'); idx2 = strfind(fn, '.');
fn1 = fn(1:idx); fn3 = fn(idx2:end);
for i = 1:10,
  fn2 = int2str(i);
  newfn = strcat(fn1, fn2);
  newfn = strcat(newfn, fn3)
end
```



function() & M files

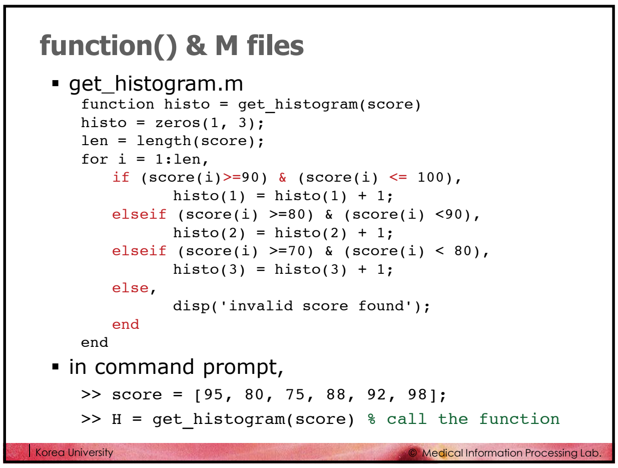
```
>> score = [95, 80, 75, 88, 92, 98];
>> avg = mean(score);
```

- You might want to repeat the same program above many times with different data sets.
  1. average.m

```
function avgval = average (grade)
  len = length(grade);
  if (isempty(len)),
    disp('error: no data in the input array');
  else,
    avgval = mean(grade);
  end
```

  2. in command window or in other M files

```
>> score = [95, 80, 75, 88, 92, 98];
>> average(score) % call the function average()
```



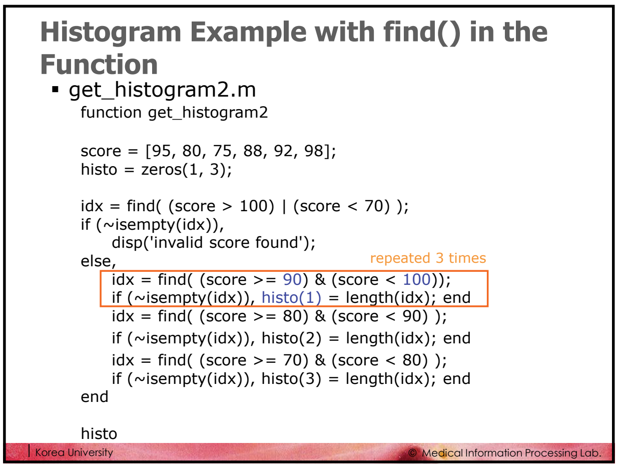
function() & M files

- get_histogram.m

```
function histo = get_histogram(score)
histo = zeros(1, 3);
len = length(score);
for i = 1:len,
    if (score(i)>=90) & (score(i) <= 100),
            histo(1) = histo(1) + 1;
    elseif (score(i) >=80) & (score(i) <90),
            histo(2) = histo(2) + 1;
    elseif (score(i) >=70) & (score(i) < 80),
            histo(3) = histo(3) + 1;
    else,
            disp('invalid score found');
    end
end
```

- in command prompt,

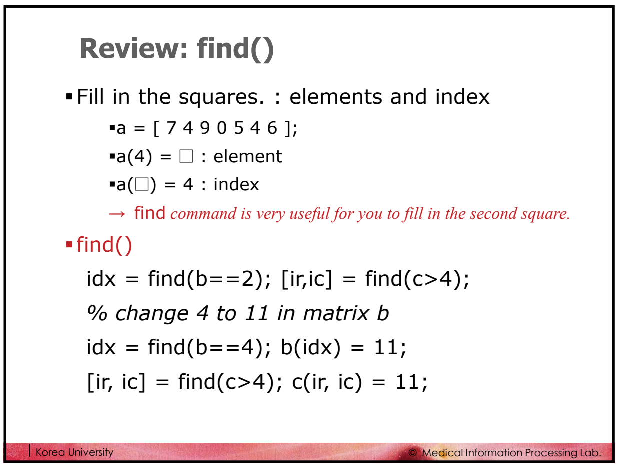
```
>> score = [95, 80, 75, 88, 92, 98];
>> H = get_histogram(score) % call the function
```



Histogram Example with find() in the Function

- get_histogram2.m

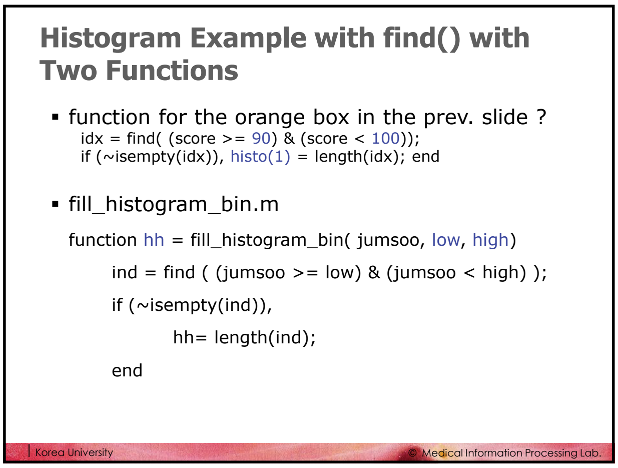
```
function get_histogram2

score = [95, 80, 75, 88, 92, 98];
histo = zeros(1, 3);

idx = find( (score > 100) | (score < 70) );
if (~isempty(idx)),
    disp('invalid score found');
else,                                        repeated 3 times
    idx = find( (score >= 90) & (score < 100));
    if (~isempty(idx)), histo(1) = length(idx); end
    idx = find( (score >= 80) & (score < 90) );
    if (~isempty(idx)), histo(2) = length(idx); end
    idx = find( (score >= 70) & (score < 80) );
    if (~isempty(idx)), histo(3) = length(idx); end
end

histo
```



Review: find()

- Fill in the squares. : elements and index
  - a = [7 4 9 0 5 4 6];
  - a(4) = □ : element
  - a(□) = 4 : index
  - → find *command is very useful for you to fill in the second square.*
- find()

```
idx = find(b==2); [ir,ic] = find(c>4);
% change 4 to 11 in matrix b
idx = find(b==4); b(idx) = 11;
[ir, ic] = find(c>4); c(ir, ic) = 11;
```



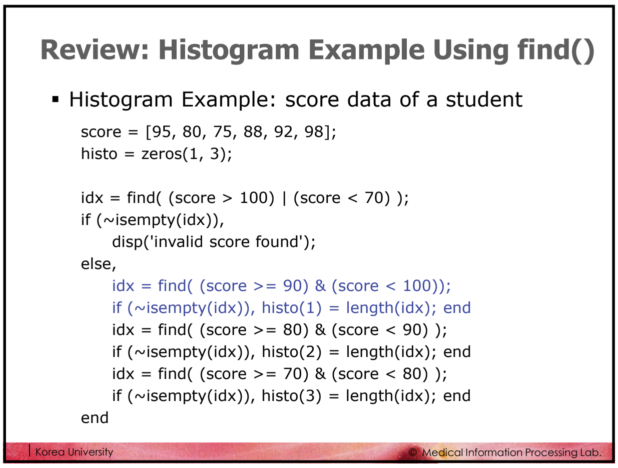
Histogram Example with find() with Two Functions

- function for the orange box in the prev. slide ?

```
idx = find( (score >= 90) & (score < 100));
if (~isempty(idx)), histo(1) = length(idx); end
```

- fill_histogram_bin.m

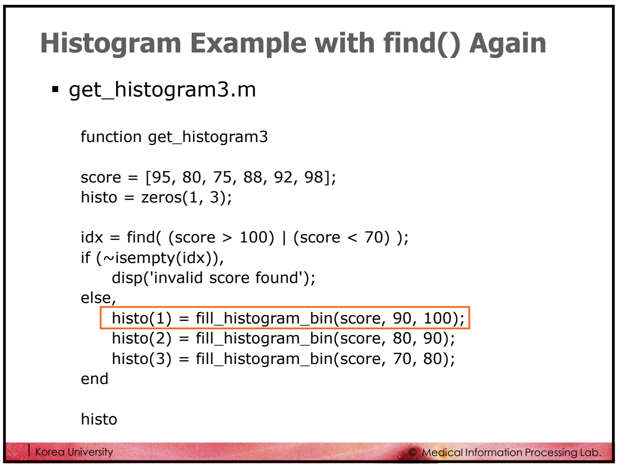
```
function hh = fill_histogram_bin( jumsoo, low, high)
    ind = find ( (jumsoo >= low) & (jumsoo < high) );
    if (~isempty(ind)),
            hh= length(ind);
    end
```



Review: Histogram Example Using find()

- Histogram Example: score data of a student

```
score = [95, 80, 75, 88, 92, 98];
histo = zeros(1, 3);

idx = find( (score > 100) | (score < 70) );
if (~isempty(idx)),
    disp('invalid score found');
else,
    idx = find( (score >= 90) & (score < 100));
    if (~isempty(idx)), histo(1) = length(idx); end
    idx = find( (score >= 80) & (score < 90) );
    if (~isempty(idx)), histo(2) = length(idx); end
    idx = find( (score >= 70) & (score < 80) );
    if (~isempty(idx)), histo(3) = length(idx); end
end
```



Histogram Example with find() Again

- get_histogram3.m

```
function get_histogram3

score = [95, 80, 75, 88, 92, 98];
histo = zeros(1, 3);

idx = find( (score > 100) | (score < 70) );
if (~isempty(idx)),
    disp('invalid score found');
else,
    histo(1) = fill_histogram_bin(score, 90, 100);
    histo(2) = fill_histogram_bin(score, 80, 90);
    histo(3) = fill_histogram_bin(score, 70, 80);
end

histo
```

Debugging

- open the M file to be debugged
  - F12 : break point
- In command prompt type the M-file name.
  - F5: start debugging
  - F10: step over
  - F11: step in
- Demo with get_histogram3.m

File Input and Output: Image Files

- read image
  - imread('filename');
  - .tif, .tiff, .jpg, .jpeg,.gif,.bmp, etc.

```
>> f = imread('d:\img\lena.jpg');
```

- display 2D image

```
>> imshow(f, G); % default G(# of intensity) = 256
>> imshow(f,[low,high]); or imshow(f,[]);
```

- save image

```
>> imwrite(f, 'd:\img\lena.bmp');
>> imwrite(f, 'd:\img\lena.jpg', 'quality' q);
% q: 0~100: The higher, the better quality.
```

  - .tif, .tiff, .jpg, .jpeg,.bmp, etc.

String: a series of characters

- initialization

```
fn = 'input1.bin'; % fn is an array storing characters
```

- how to make 'input2.bin' using 'input1.bin'

```
idx = strfind(fn,'t'); % help strfind for more details
idx = idx + 1;
newfn = fn;
newfn(idx) = int2str(2);
```

- how to make filenames from 'input1.bin' to 'input10.bin'

```
idx = strfind(fn,'t'); idx2 = strfind(fn, '.');
fn1 = fn(1:idx); fn3 = fn(idx2:end);
for i = 1:10,
  fn2 = int2str(i);
  newfn = strcat(fn1, fn2);
  newfn = strcat(newfn, fn3)
end
```

Hands-on

File Input and Output: Matrix Data

```
>> a = 1:10;
```

- Save matrix data into file

```
>> save('a.mat', 'a'); % matlab matrix format
>> save('a.txt', 'a', '-ascii'); % text file
```

- Read matrix data from file

```
% read from the matlab binary format file
>> amat = load('a.mat');
% read from the standard text file
>> atxt = load('a.txt'); % text file
```