INTRODUCTION TO NUMERICAL ANALYSIS

Cho, Hyoung Kyu

Department of Nuclear Engineering

Seoul National University

0. MATLAB USAGE

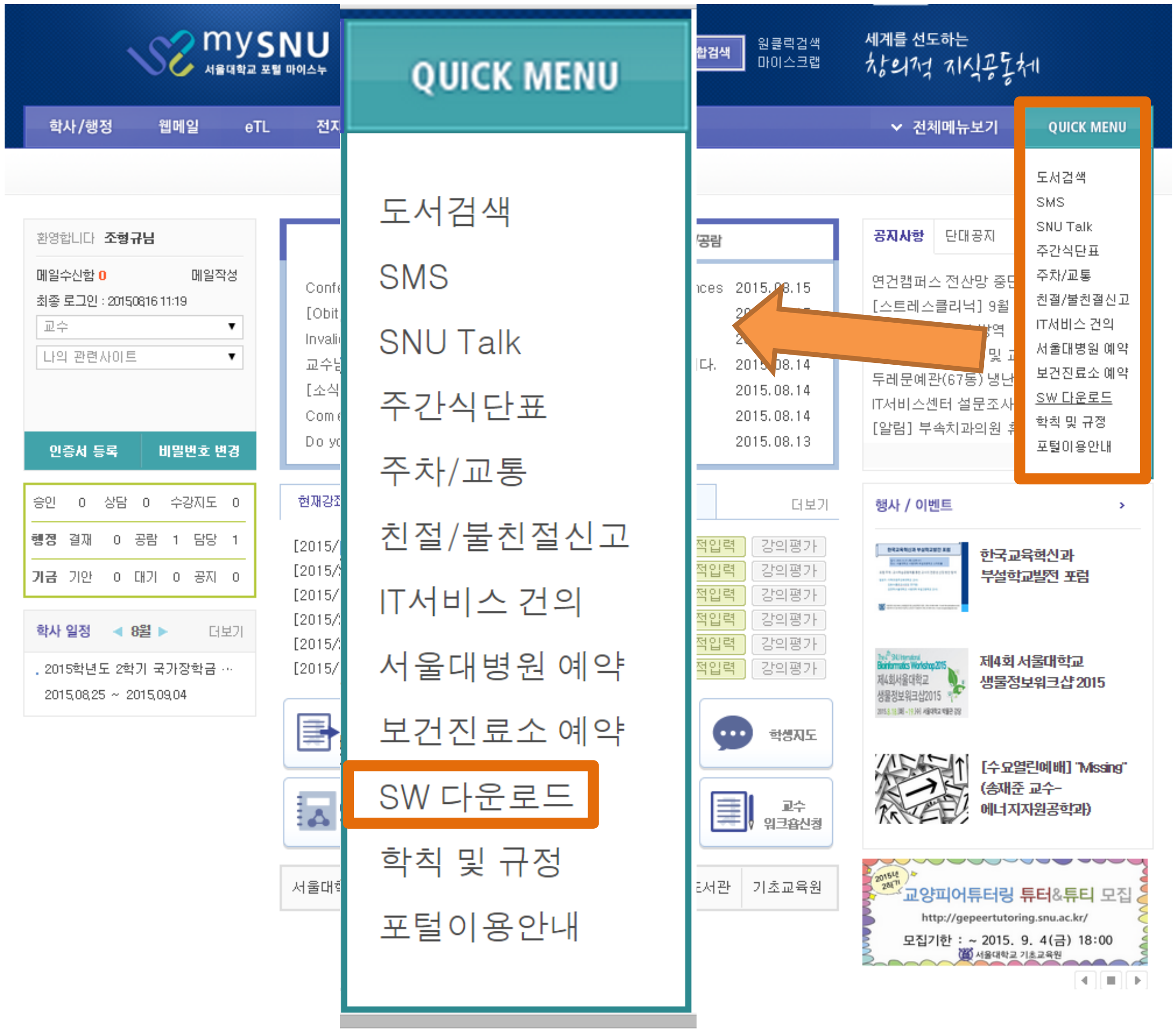
MATLAB

- **MAT**rix **LAB**oratory
- Mathematical computations, modeling and simulations, data analysis and processing, visualization and graphics, and algorithm development

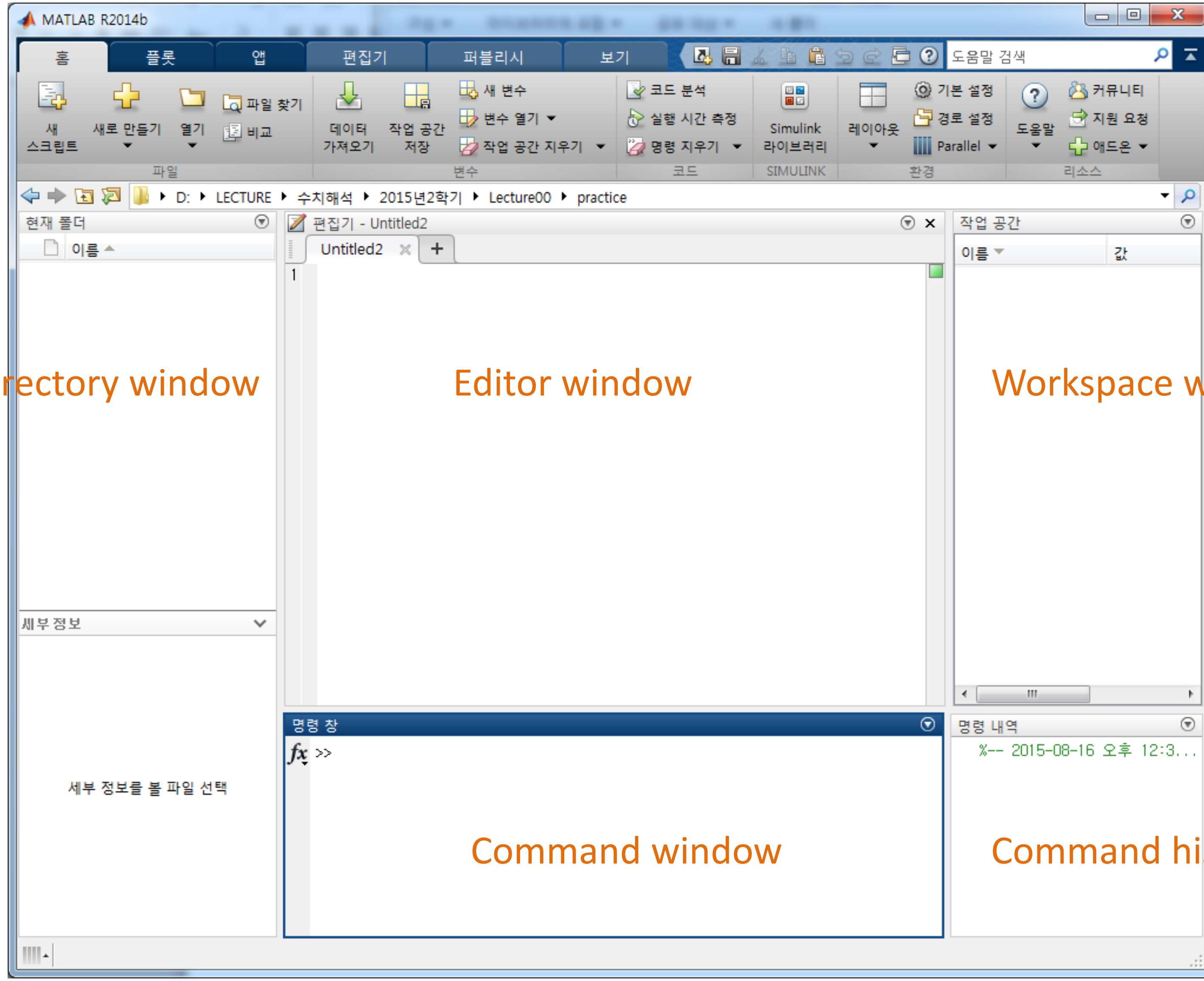
- Standard MATLAB
- Toolboxes
  - Signal processing, symbolic calculations, control system, image processing, etc.

2. Starting with MATLAB

❖ MATLAB download



❖ MATLAB interface



2. Starting with MATLAB

❖ Command window

- Commands are typed next to the prompt (>>) and are executed when the *Enter* key is pressed.

- Output generated by the command is displayed in the Command Window, unless a semicolon (;) is typed at the end.

- *clc* command: clear Command Window
- Up-arrow and down-arrow keys (↑ and ↓): recall commands typed before

Operation	Symbol	Example	Operation	Symbol	Example
Addition	+	5 + 3	Right division	/	5 / 3
Subtraction	–	5 – 3	Left division	\	5 \ 3 = 3 / 5
Multiplication	*	5 * 3	Exponentiation	^	5 ^ 3 (means $5^3 = 125$)

```
>> 7 + 8/2
ans =
   11
>> (7+8)/2 + 27^(1/3)
ans =
   10.5000
```

Command window

- Numerical values can be assigned to variables.

```
>> a = 12
a =
   12
>> B = 4;
>> C = (a - B) + 40 - a/B*10
C =
   18
```

Since a semicolon is typed at the end of the command, the value of B is not displayed.

- Elementary math built-in functions
  - Ex) *sqrt(x)*

```
>> sqrt(64)
ans =
   8
>> sqrt(50 + 14*3)
ans =
   9.5917
```

```
>> sqrt(54 + 9*sqrt(100))
ans =
   12
>> (15 + 600/4)/sqrt(121)
ans =
   15
```

Argument includes a function.

Function is included in an expression.

Built-in elementary math functions

Command	Description	Example

`sqrt(x)`	Square root.	`>> sqrt(81)`
		`ans =`
		`9`

`exp(x)`	Exponential (e^x).	`>> exp(5)`
		`ans =`
		`148.4132`

`abs(x)`	Absolute value.	`>> abs(-24)`
		`ans =`
		`24`

`log(x)`	Natural logarithm.
	Base e logarithm (ln).
		`>> log(1000)`
		`ans =`
		`6.9078`

`log10(x)`	Base 10 logarithm.	`>> log10(1000)`
		`ans =`
		`3.0000`

`sin(x)`
 	
`sind(x)`
	Sine of angle x (x in radians).
	
	Sine of angle x (x in degrees).
		`>> sin(pi/6)`
		`ans =`
		`0.5000`
		
		`>> sind(30)`
		`ans =`
		`0.5000`

Built-in elementary math functions

Command	Description	Example
The other trigonometric functions are written in the same way. The inverse trigonometric functions are written by adding the letter "a" in front, for example, `asin(x)`.		

`round(x)`	Round to the nearest integer.	`>> round(17/5)`
		`ans =`
		`3`

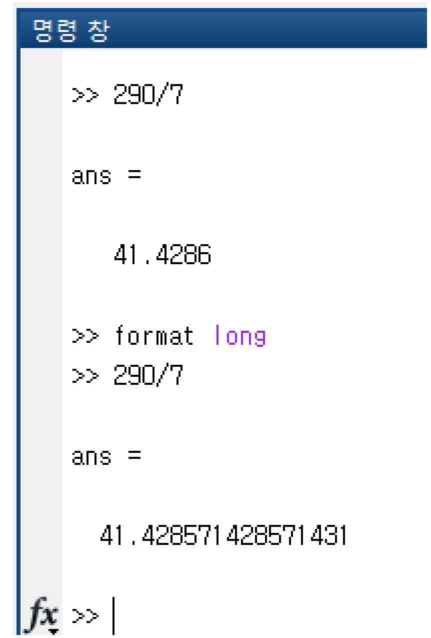
`fix(x)`	Round toward zero.	`>> fix(9/4)`
		`ans =`
		`2`
		`>> fix(-9/4)`
		`ans =`
		`-2`

`ceil(x)`	Round up toward infinity.	`>> ceil(11/5)`
		`ans =`
		`3`

`floor(x)`	Round down toward minus infinity.	`>> floor(-9/4)`
		`ans =`
		`-3`

Display format

Command	Description	Example
`format short`	Fixed point with four decimal digits for: $0.001 \le number \le 1000$ Otherwise display format `short e`.	`>> 290/7` `ans =` `41.4286`
`format long`	Fixed point with 14 decimal digits for: $0.001 \le number \le 100$ Otherwise display format `long e`.	`>> 290/7` `ans =` `41.42857142857143`
`format short e`	Scientific notation with four decimal digits.	`>> 290/7` `ans =` `4.1429e+001`
`format long e`	Scientific notation with 15 decimal digits.	`>> 290/7` `ans =` `4.142857142857143e+001`
`format short g`	Best of 5-digit fixed or floating point.	`>> 290/7` `ans =` `41.429`
`format long g`	Best of 15-digit fixed or floating point.	`>> 290/7` `ans =` `41.4285714285714`
`format bank`	Two decimal digits.	`>> 290/7` `ans =` `41.43`



3. Arrays

Arrays

- 1D array: vector
- 2D array: matrix
- Each number in a vector or a matrix: element

- Creating a vector in MATLAB

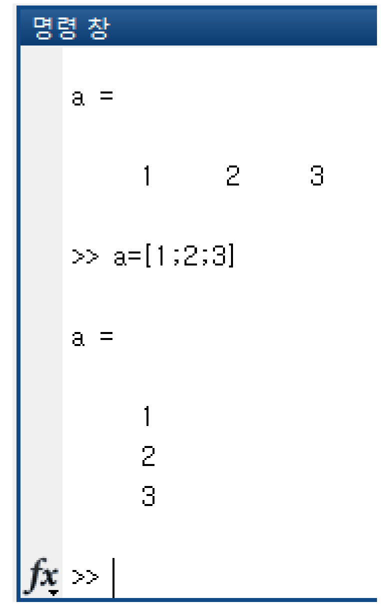
```
variable_name = [number number ... number]
```

Row vector

variable_name=[number; number; number]

Column vector

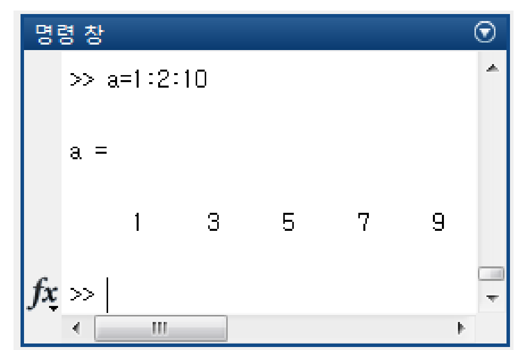
  - ";" is placed!




```
variable_name = m:q:n
```

Row vector with constant spacing

  - m : first element
  - q : spacing
  - n : last element



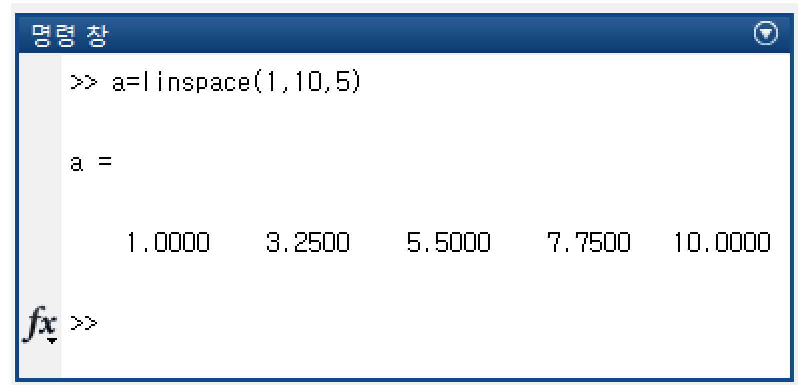
❖ Arrays

- Creating a vector in MATLAB

  "linspace" command

```
variable_name = linspace(xi,xf,n)
```

  - xi : first element
  - xf : final element
  - n : number of elements



Arrays

- Creating a matrix in MATLAB

```
variable_name = [1st row elements; 2nd row ele-
        ments; .... ; last row elements]
```

  - Variables and mathematical expressions can be used as elements.

명령 창

```
>> a= [5 35 43; 4 76 81; 21 32 40]

a =

     5    35    43
     4    76    81
    21    32    40

fx >>
```

명령 창

```
>> cd = 6; e = 3; h = 4;
>> Mat= [e, cd*h, cos(pi/3); h^2, sqrt(h*h/cd), 14 ]

Mat =

    3.0000   24.0000    0.5000
   16.0000    1.6330   14.0000

fx >>
```

- Array addressing

$$ve = 35 \ \ 46 \ \ 78 \ \ 23 \ \ 5 \ \ 14 \ \ 81 \ \ 3 \ \ 55$$

$$ve(4) = 23,\ ve(7) = 81, \text{ and } ve(1) = 35$$

$$ma = \begin{bmatrix} 3 & 11 & 6 & 5 \\ 4 & 7 & 10 & 2 \\ 13 & 9 & 0 & 8 \end{bmatrix} \qquad ma(1,1) = 3, \text{ and } ma(2,3) = 10$$

❖ Arrays

- Array addressing

```
>> VCT = [35  46  78  23  5  14  81  3  55]        Define a vector.
VCT =
   35    46    78    23     5    14    81     3    55
>> VCT(4)=-2; VCT(6)=273        Assign new values to the fourth and sixth elements.
VCT =
   35    46    78    -2     5   273    81     3    55
>> VCT(5)^VCT(8) + sqrt(VCT(7))        Use vector elements in a mathematical expression.
ans =
   134
>> MAT = [3 11 6 5; 4 7 10 2; 13 9 0 8]        Define a matrix.
MAT =
     3    11     6     5
     4     7    10     2
    13     9     0     8
>> MAT(3,1) = 20        Assign a new value to the (3,1) element.
MAT =
     3    11     6     5
     4     7    10     2
    20     9     0     8
>> MAT(2,4) - MAT(1,2)        Use matrix elements in a mathematical expression.
ans =
    -9
```

Arrays

- Array addressing
  - Using a colon ": " in addressing arrays

```
>>v=[4 15 8 12 34 2 50 23 11]
v =
   4    15     8    12    34     2    50    23    11
>>u=v(3:7)
u =
   8    12    34     2    50
>>A=[1 3 5 7 9 11; 2 4 6 8 10 12; 3 6 9 12 15 18; 4 8 12 16 20
24; 5 10 15 20 25 30]
A =
     1     3     5     7     9    11
     2     4     6     8    10    12
     3     6     9    12    15    18
     4     8    12    16    20    24
     5    10    15    20    25    30
C = A(2,:)
C =
     2     4     6     8    10    12
>> F = A(1:3,2:4)
```

Define a vector.

Vector u is created from the elements 3 through 7 of vector v.

Define a matrix.

Vector C is created from the second row of matrix A.

Matrix F is created from the elements in rows 1 through 3 and columns 2 through 4 of matrix A.

3. Arrays

❖ Arrays

- Built-in functions for handling arrays

Command	Description	Example
length(A)	Returns the number of elements in vector A.	>> A = [5 9 2 4]; >> length(A) ans = 4

size(A)	Returns a row vector [m,n], where m and n are the size $m \times n$ of the array A. (m is number of rows. n is number of columns.)	>> A = [6 1 4 0 12; 5 19 6 8 2]
		A =
		  6 1 4 0 12
		  5 19 6 8 2
		>> size(A)
		ans =
		  2 5

zeros(m,n)	Creates a matrix with *m* rows and *n* columns, in which all the elements are the number 0.	>> zr = zeros(3,4)
		zr =
		  0 0 0 0
		  0 0 0 0
		  0 0 0 0

ones(m,n)	Creates a matrix with *m* rows and *n* columns, in which all the elements are the number 1.	>> ne = ones(4,3)
		ne =
		  1 1 1
		  1 1 1
		  1 1 1
		  1 1 1

eye(n)	Creates a square matrix with *n* rows and *n* columns in which the diagonal elements are equal to 1 (identity matrix).	>> idn = eye(3)
		idn =
		  1 0 0
		  0 1 0
		  0 0 1

size(A,1)
size(A,2)

❖ Strings

- A string is an array of characters.
- It is created by typing the characters within single quotes.
- Strings can include letters, digits, other symbols, and spaces.

- Concatenate strings horizontally
  - strcat : string concatenate!
  - How can create ‘Nuclear Engineering’ using strcat command?
  - 32 ?

명령 창

```
>> a='Nuclear Engineering'

a =

Nuclear Engineering

>> a(1:7)

ans =

Nuclear
```

명령 창

```
>> a='Nuclear';
>> b='Engineering';
>> strcat(a,b)

ans =

NuclearEngineering

>> strcat(a,32,b)

ans =

Nuclear Engineering

fx >>
```

4. Mathematical Operations with Arrays

❖ Addition and subtraction of arrays

```
>> VA = [8 5 4]; VB = [10 2 7];          Define two vectors VA and VB.
>> VC = VA + VB                           Define a vector VC that is equal to VA + VB.
VC =
  18    7   11
>> A = [5 -3 8; 9 2 10], B = [10 7 4; -11 15 1]     Define two matrices A and B.
A =
     5   -3    8
     9    2   10
B =
    10    7    4
   -11   15    1
>> C = A + B                              Define a matrix C that is equal to A + B.
C =
    15    4   12
    -2   17   11
>> C - 8                                  Subtract 8 from the matrix C.
ans =                                     8 is subtracted from each element of C.
     7   -4    4
   -10    9    3
```

4. Mathematical Operations with Arrays

❖ Multiplications of arrays

```
>> A = [2 -1; 8 3; 6 7], B = [4 9 1 -3; -5 2 4   Define two matrices A and B.
A =
    2   -1
    8    3
    6    7
B =
    4    9    1   -3
   -5    2    4    6
>> C = A*B                                       Multiply A*B.
C =                                              C is a (3 × 4) matrix.
   13   16   -2  -12
   17   78   20   -6
  -11   68   34   24
```

4. Mathematical Operations with Arrays

- **Array division ("/")**

$[a][x] = [b]$ → $\boxed{x = a\backslash b}$ $[x][a] = [b]$ → $\boxed{x = b/a}$

- **Element-by-element operations**

Symbol	Description	Symbol	Description
.*	Multiplication	./	Right division
.^	Exponentiation	.\	Left Division

```
>> A = [2  6  3;  5  8  4]
A =
   2    6    3
   5    8    4
>> B = [1  4  10;  3  2  7]
B =
   1    4   10
   3    2    7
>> A .* B
ans =
   2   24   30
  15   16   28
```

Define a (2×3) matrix A.

Define a (2×3) matrix B.

Element-by-element multiplication of arrays A and B.

Element-by-element operations

```
>> C = A ./ B                    Element-by-element division of array A by array B.
C =
   2.0000    1.5000    0.3000
   1.6667    4.0000    0.5714
>> B .^ 3                        Element-by-element exponentiation of array B.
ans =
     1    64   1000
    27     8    343
```

- Usefulness of element-by-element operation
  - For example,

```
>> z = [1:2:15]                  Define a vector z with eight elements.
z =
   1    3    5    7    9   11   13   15
>> y = (z.^3 + 5*z)./(4*z.^2 - 10)    Vector z is used in element-by-element
                                      calculation of the elements of vector y.
y =
  -1.0000   1.6154   1.6667   2.0323   2.4650   2.9241   3.3964   3.8764
```

4. Mathematical Operations with Arrays

Built-in functions for handling arrays

Command	Description	Example

`mean(A)`	If A is a vector, the function returns the mean value of the elements of the vector.	`>> A=[5 9 2 4];`
		`>> mean(A)`
		`ans =`
		`5`

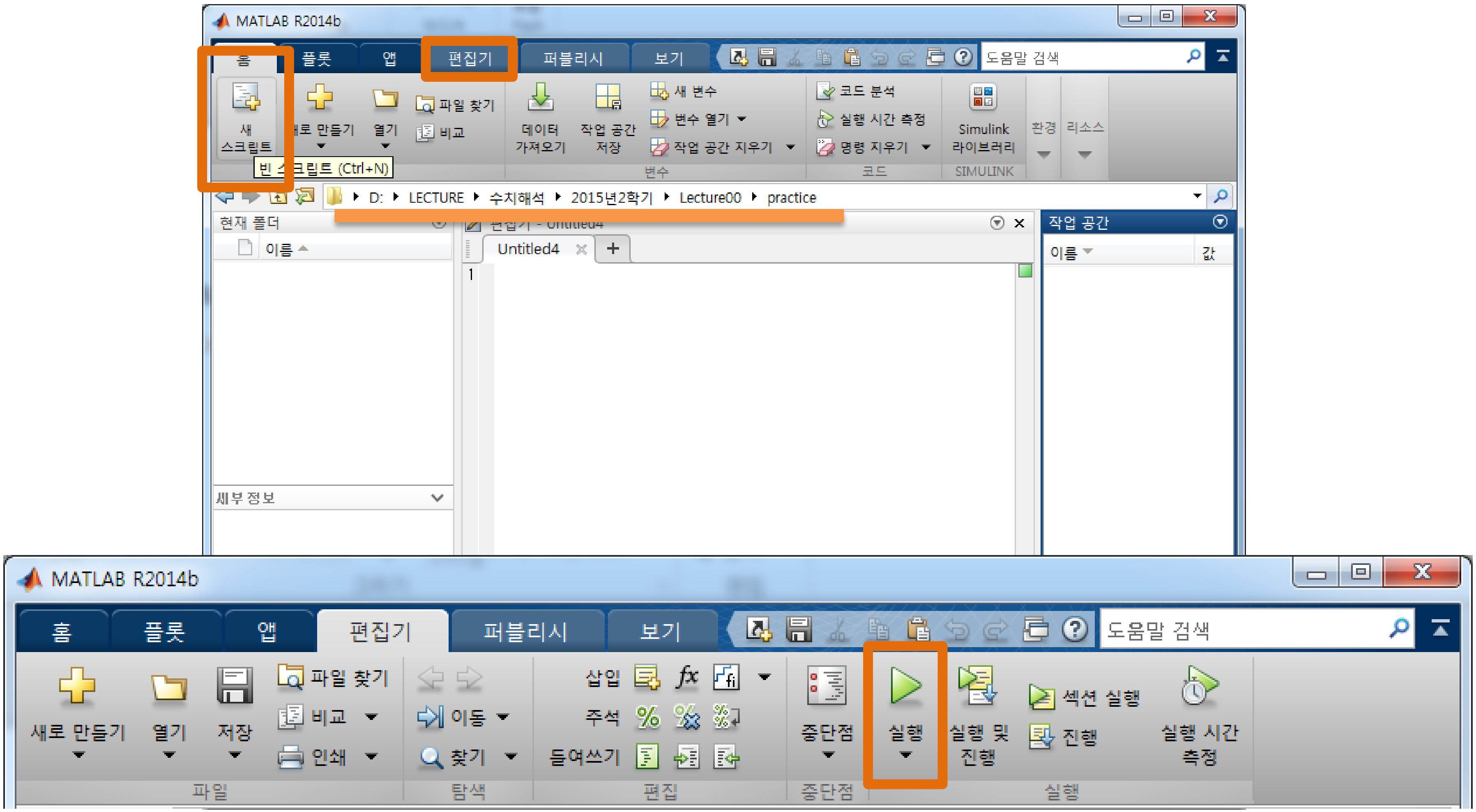
`sum(A)`	If A is a vector, the function returns the sum of the elements of the vector.	`>> A=[5 9 2 4];`
		`>> sum(A)`
		`ans =`
		`20`

`sort(A)`	If A is a vector, the function arranges the elements of the vector in ascending order.	`>> A=[5 9 2 4];`
		`>> sort(A)`
		`ans =`
		`2 4 5 9`

`det(A)`	The function returns the determinant of a square matrix A.	`>> A=[2 4; 3 5];`
		`>> det(A)`
		`ans =`
		`-2`

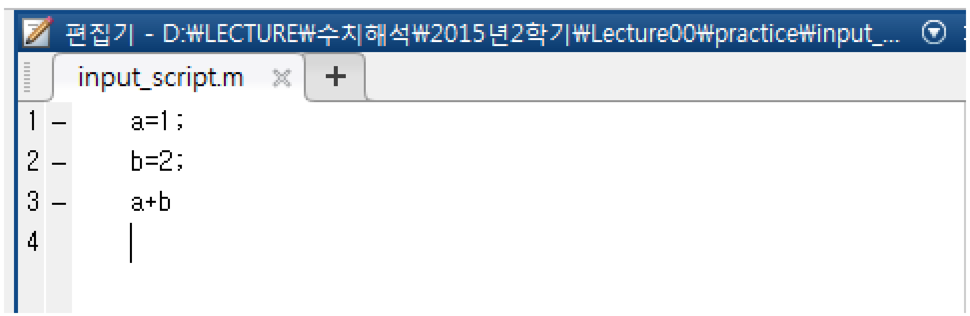
Script file

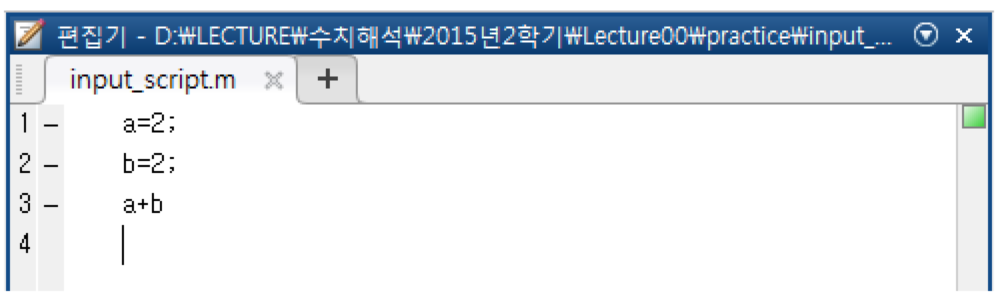
- Program: a file that contains a sequence of MATLAB commands
- Extension: ".m" $\Rightarrow$ M-file

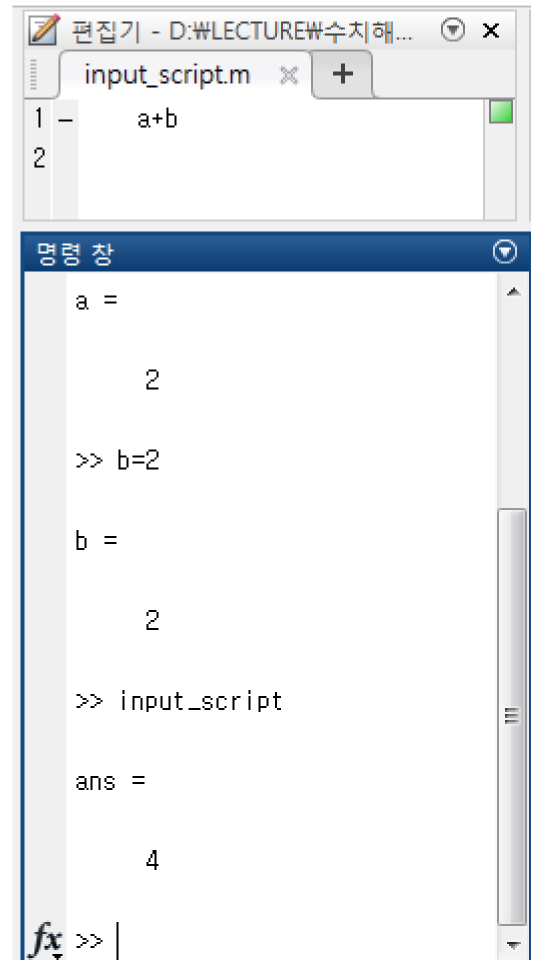


Script file

- Input to a script file
  - Define the variable and assign it a value in the script file
    - File must be edited and the assignment of the variable changed.
  - Define the variable and assign it a value in the Command Window

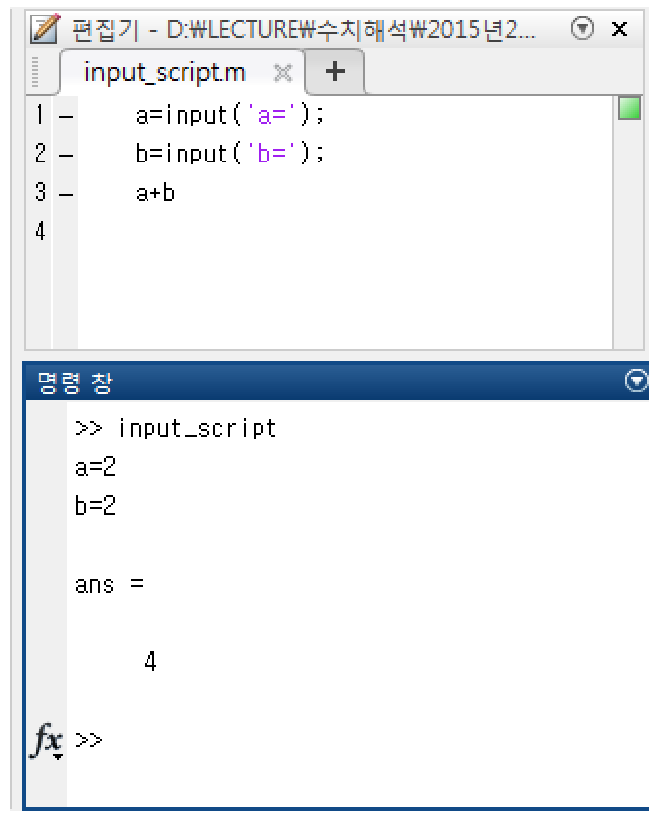






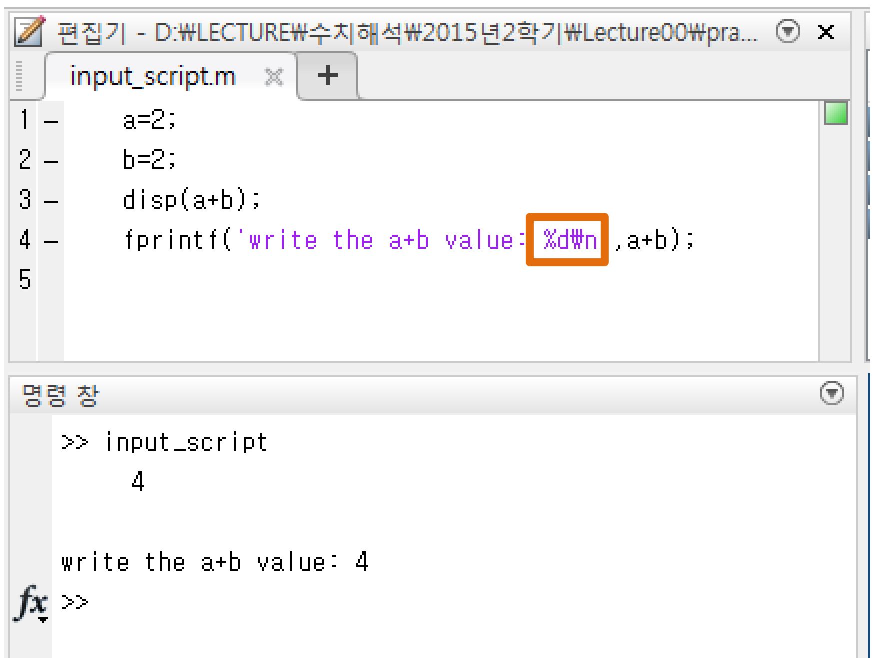
Script file

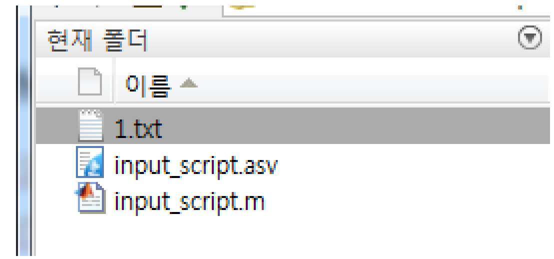
- Input to a script file
  - Define the variable in the script file but assign a specific value in the Command Window when the script file is executed.
  - “input” command

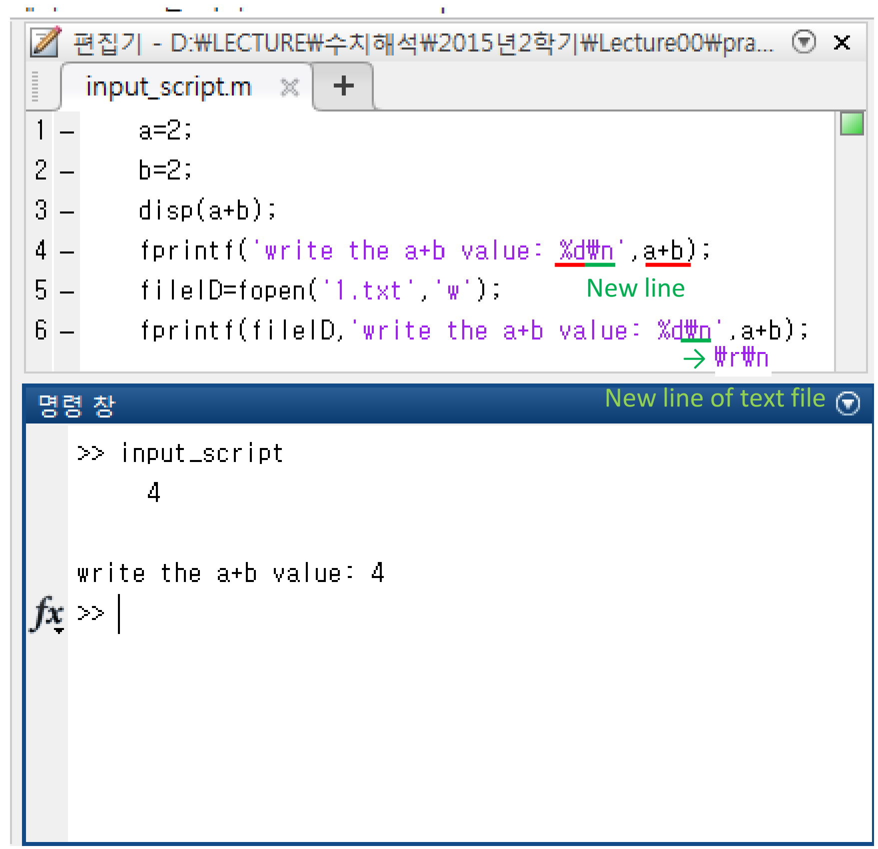


❖ Script file

- Output from a script file
  - "`disp`" command
  - "`fprintf`" command
    - To Command Window
    - To a file



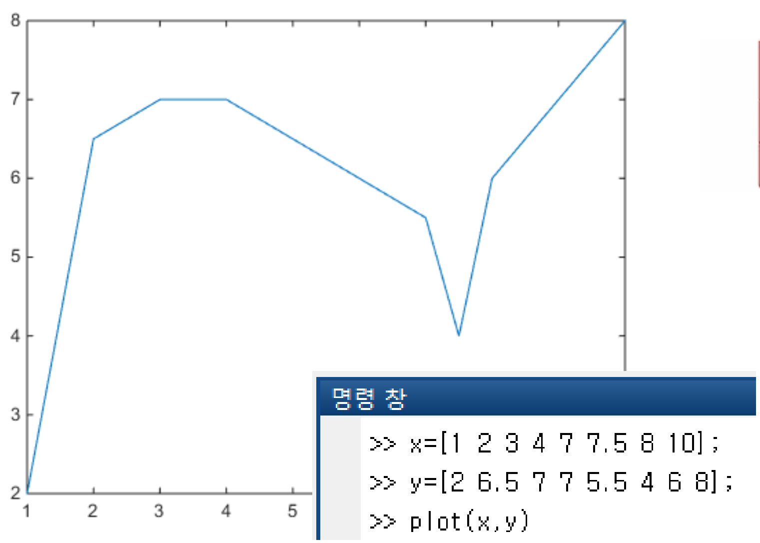




Plots in MATLAB

- Standard plots with linear axes and logarithmic axes
- Bars and stairs plots
- “`plot`” command
  - With ‘line specifiers’

```
plot(x,y)
```



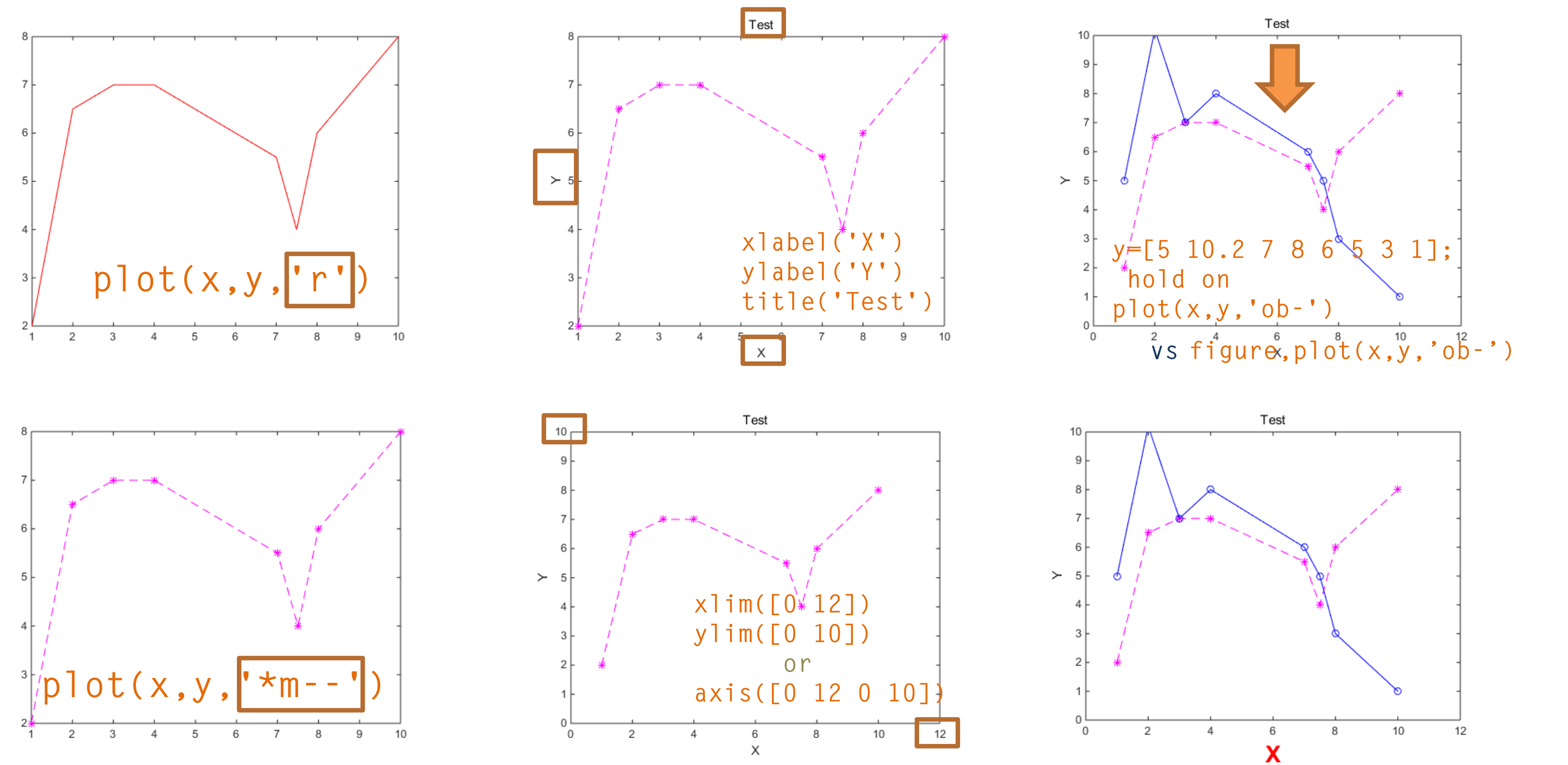
Line Color	Specifier	Line Style	Specifier	Line Color	Specifier	Line Color	Specifier
red	r	blue	b	magenta	m	black	k
green	g	cyan	c	yellow	y	white	w

Line Style	Specifier	Line Style	Specifier
solid (default)	-	dotted	:
dashed	--	dash-dot	-.

Marker	Specifier	Marker	Specifier	Marker	Specifier
plus sign	+	asterisk	*	square	s
circle	o	point	.	diamond	d

Plots in MATLAB

- "plot" command
  - With 'line specifiers', labels, title

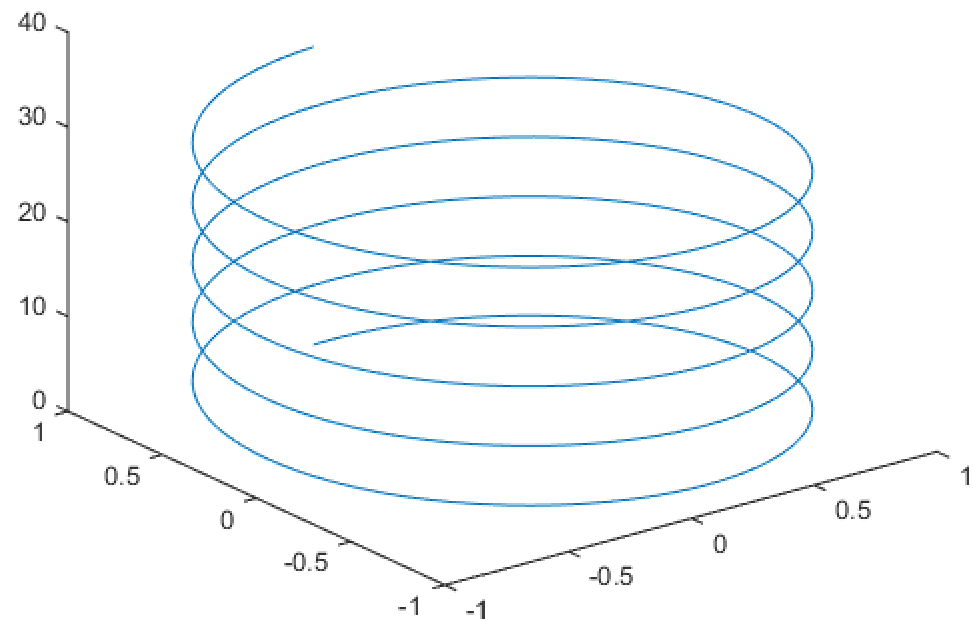



```
xlabel('X','FontSize',20, …
'FontWeight','bold','Color','r')
```

Plots in MATLAB

- "plot3" command

```
>> t = 0:pi/50:10*pi;
>> st = sin(t);
>> ct = cos(t);

>> figure
>> plot3(st,ct,t)
```



❖ Plots in MATLAB

- “surf” command

Meshgrid(-1:0.5:1);

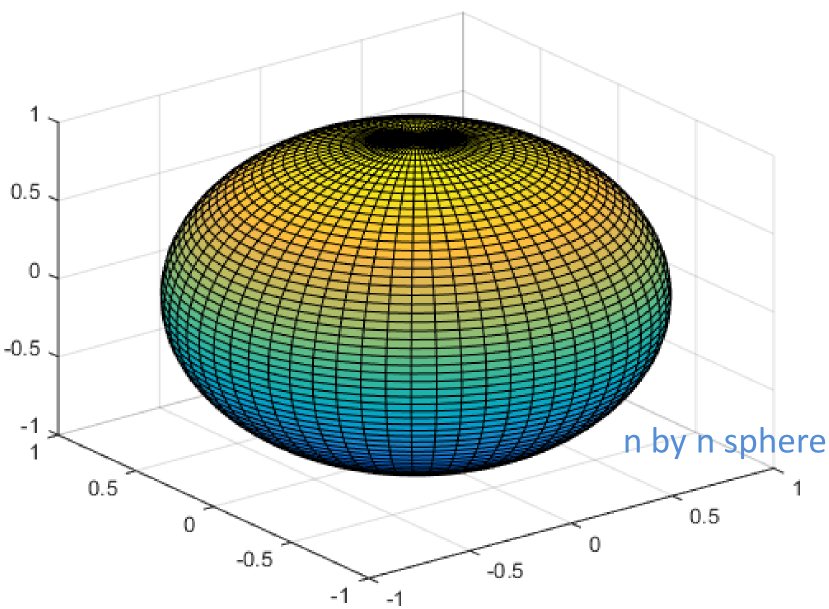
x — 5x5 double

	1	2	3	4	5
1	-1	-0.5000	0	0.5000	1
2	-1	-0.5000	0	0.5000	1
3	-1	-0.5000	0	0.5000	1
4	-1	-0.5000	0	0.5000	1
5	-1	-0.5000	0	0.5000	1

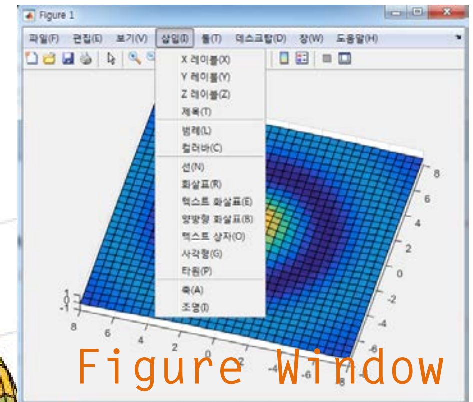
y — 5x5 double

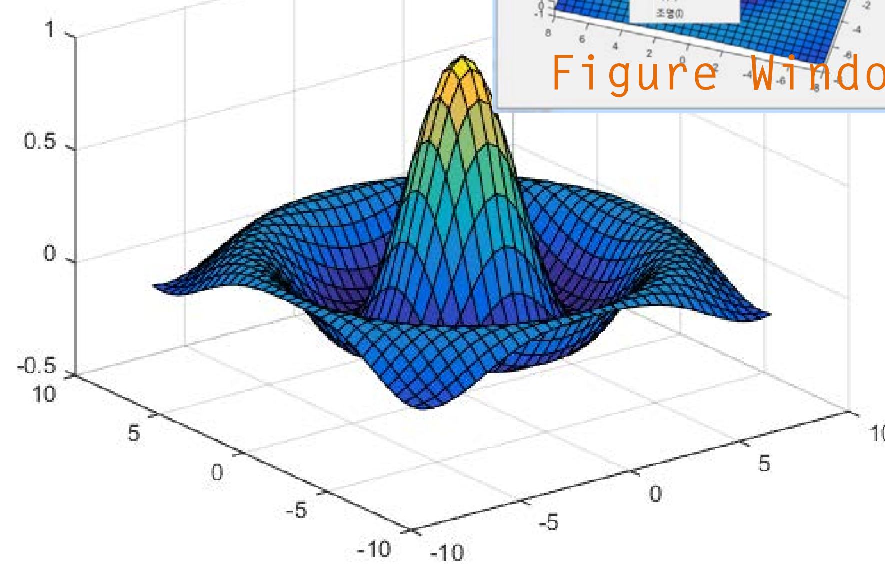
	1	2	3	4	5
1	-1	-1	-1	-1	-1
2	-0.5000	-0.5000	-0.5000	-0.5000	-0.5000
3	0	0	0	0	0
4	0.5000	0.5000	0.5000	0.5000	0.5000
5	1	1	1	1	1

```
>> k = 6;
>> n = 2^k-1;
>> [x,y,z] = sphere(n);
>> figure
>> surf(x,y,z);
```



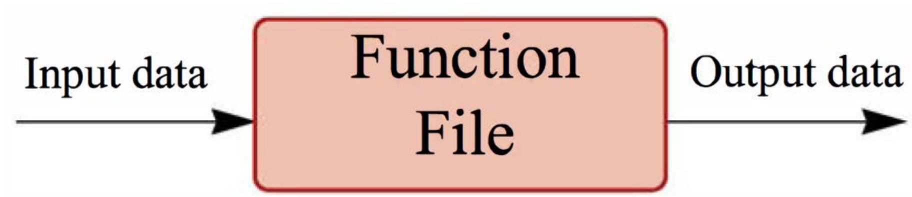

```
[x,y]=meshgrid(-8:0.5:8);
r=sqrt(x.^2+y.^2)+eps;
z=sin(r)./r;
surf(x,y,z);
```



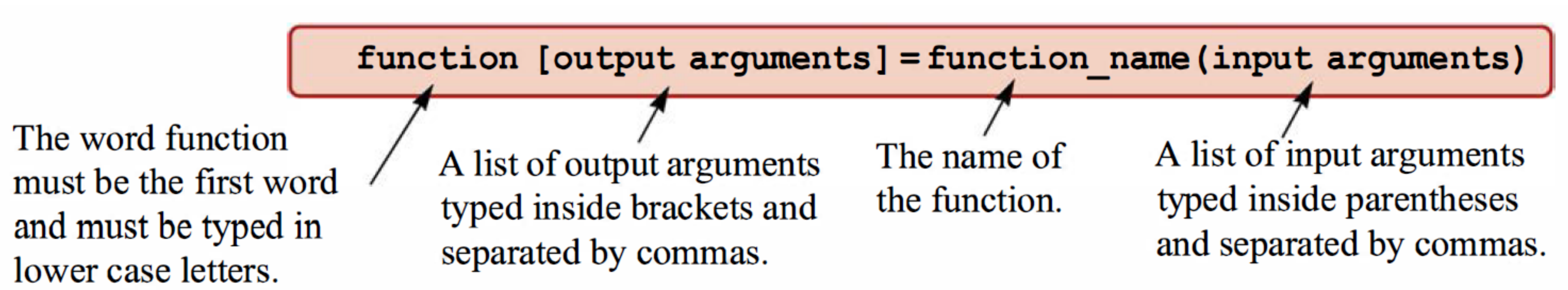


User defined function

- Arguments



- To create user-defined function
  - Similar with script files



User defined function

- Simple example of function
  - test.m

```
function [result1 result2]=test(a)
result1=a+100;
result2=a+200;
```

  - Command Window

```
[a b]=test(100)

a =

   200

b =

   300
```

Anonymous Functions

- Users can define functions in the Command Window, within a script file or inside a user-defined function

$$\boxed{\texttt{name = @ (arglist) expr}}$$

The name of the anonymous function.

The @ symbol.

A list of input arguments (independent variables).

Mathematical expression.

```
cube= @ (x) x^3
circle=@ (x, y) 16*x^2+9*y^2
parabola= @ (x) a*x^2+b*x+c
```

$$f(x) = \frac{e^{x^2}}{\sqrt{x^2+5}}$$

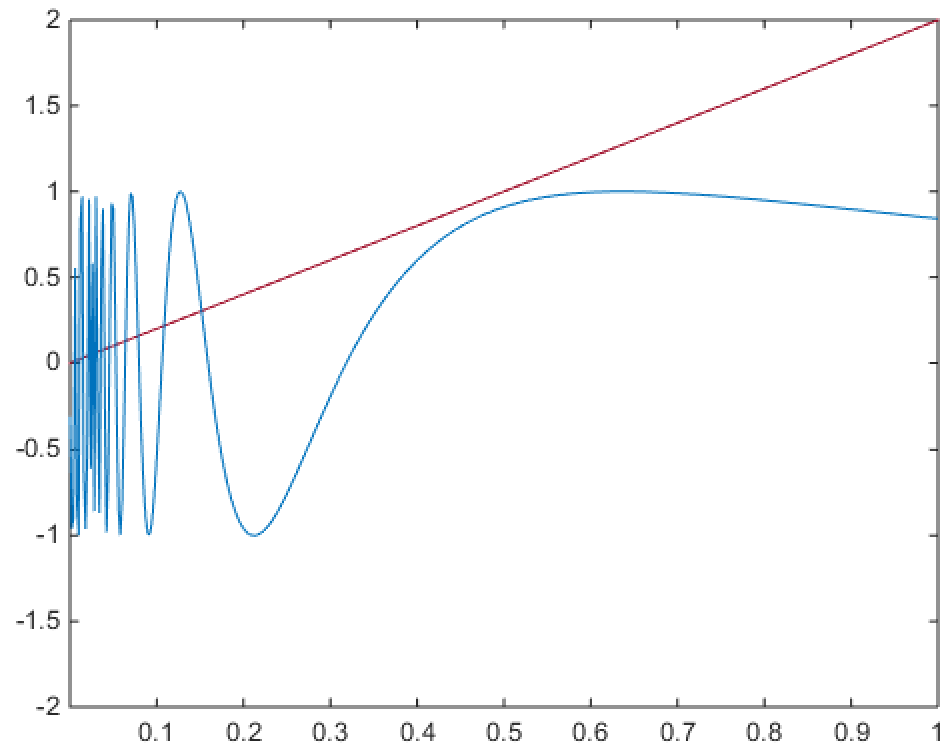
```
FA= @ (x) exp(x^2)/sqrt(x^2+5)
```

```
>> FA(2)
ans =
   18.1994
>> z=FA(3)
z =
   2.1656e+03
```

Function function

- Function that accepts a function handle as an input argument
- Function handle
  - Ex) cosHandle=@cos

```
fhandle=@(x)(2*x);
fplot(fhandle,[0.0,1.0])
hold on
fhandle=@(x)(sin(1.0/x));
fplot(fhandle,[0.0001,1.0])
ylim([-2,2])
```



Relational and logical operators

Relational Operator	Description	Relational Operator	Description
<	Less than	>=	Greater than or equal to

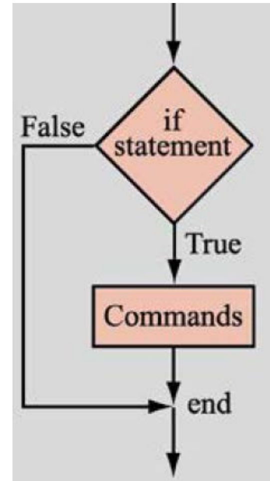
Logical operator	Name	Description
& Example: A&B	AND	Operates on two operands (A and B). If both are true, the result is true (1); otherwise the result is false (0).
| Example: A|B	OR	Operates on two operands (A and B). If either one, or both are true, the result is true (1); otherwise (both are false) the result is false (0).
~ Example: ~A	NOT	Operates on one operand (A). Gives the opposite of the operand. True (1) if the operand is false, and false (0) if the operand is true.

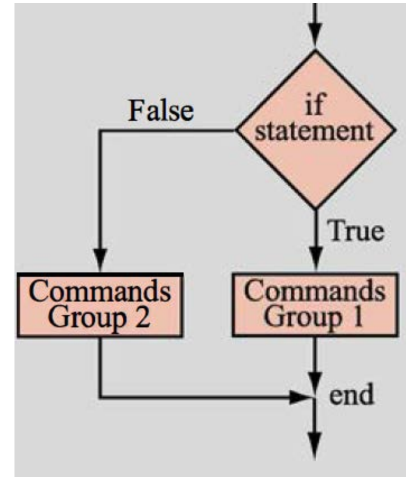
```
>> 4==5    ans = 0

>> 5>10    ans = 0

>> 0||1    ans = 1

>> 0&1     ans = 0

>> 1&5     ans = 1

>> 1||2    ans = 1

>> ~(1||0) ans = 0
```

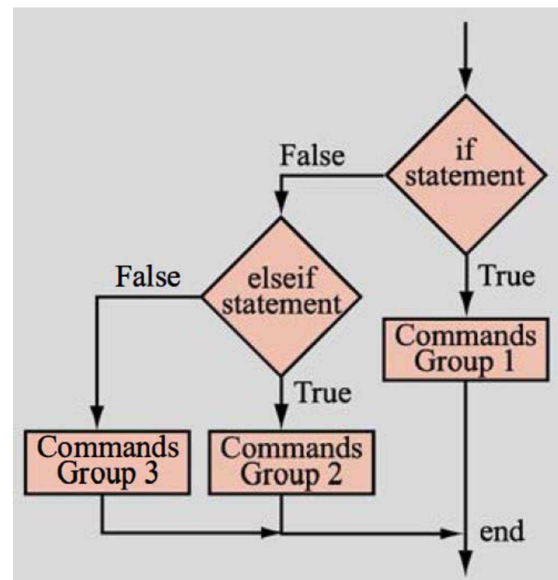
Conditional Statements, if-else Structures

- `if-end`
- `if-else-end`
- `if-elseif-else-end`
- `if-elseif-elseif…-else-end`
- In a script file,

```
function test(a)

if(a<10)
    'less than 0'
elseif(a<10)
    'greater than 0, less than 10'
elseif(a<100)
    'greater than 10, less than 100'
else
    'greater than or equal to 100'
end
```



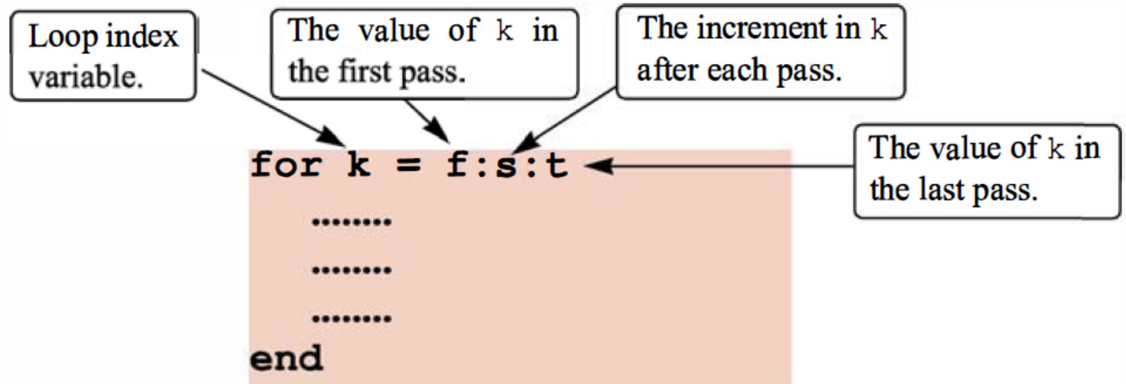




Loops

- for-end

```
function circle(r)
n=100;
for i=1:n
    theta=2*pi/n*i;
    x(i)=r*cos(theta);
    y(i)=r*sin(theta);
end

plot(x,y)
```



- 원을 쌓아 구를 그려보길.