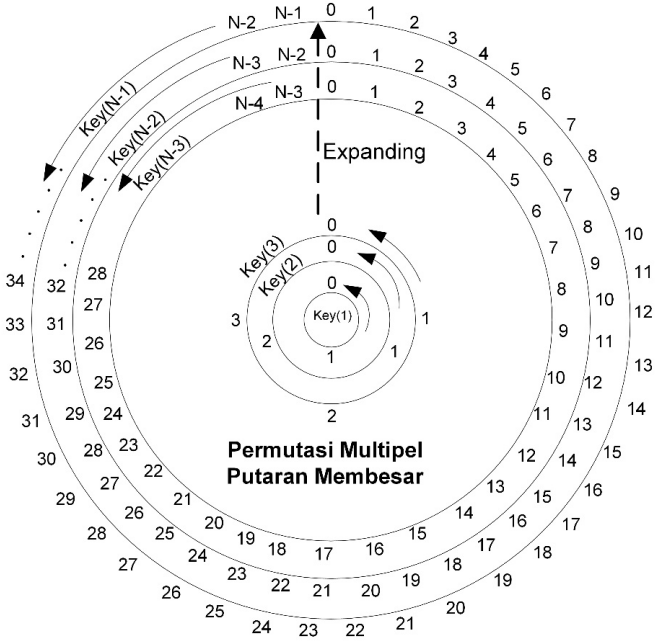


Dr. Yohan Suryanto, S.T, M.T

Semester Ganjil 2020/2021

KRIPTOGRAFI 1
Pengenalan Matlab

Outline

1. Basics
2. Visualization
3. Open Image File

1 Part I: Basics

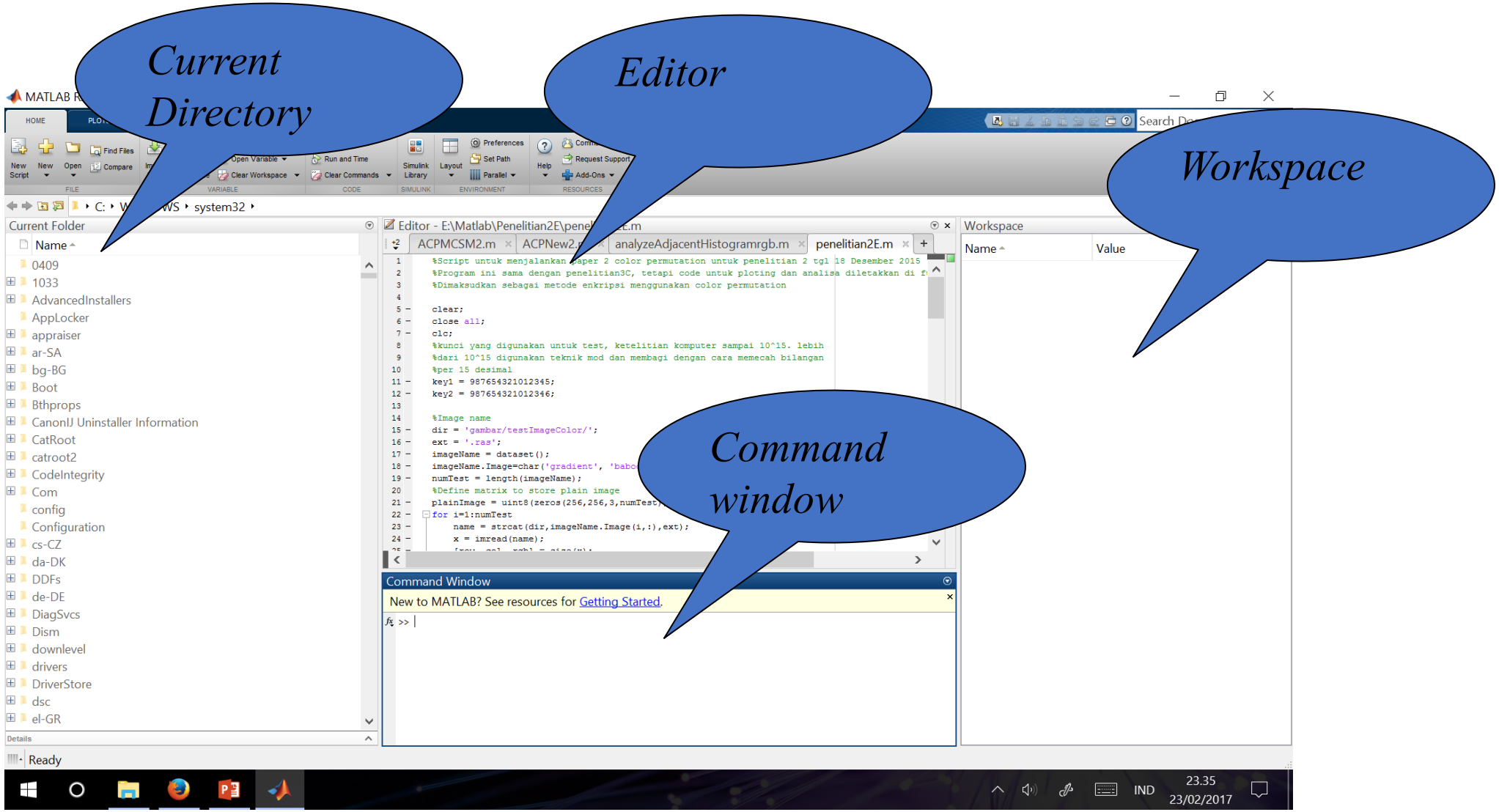
Matlab

- Stands for MATrix LABoratory
- Interpreted language
- Scientific programming environment
- Very good tool for the manipulation of matrices
- Great visualisation capabilities
- Loads of built-in functions
- Easy to learn and simple to use
- Alternative opensource: SCILAB, GNU Octave

Basic

- MATLAB Environment
- Getting Help
- Variables
- Vectors, Matrices, and Linear Algebra
- Flow Control / Loops

Display Windows



Display Windows Cont…

- Graphic (Figure) Window
  - Displays plots and graphs
    - E.g: surf(magic(30))
  - Created in response to graphics commands.

- M-file editor/debugger window
  - Create and edit scripts of commands called M-files.

Getting Help

- type one of following commands in the command window:
  - **help** – lists all the help topic
  - **help** *command* – provides help for the specified command
    - **help help** – provides information on use of the help command
  - ***Google****… of course*

Workspace

- The workspace is Matlab's memory
- Can manipulate variables stored in the workspace

```
>> b=10;
>> c=a+b
c =
    22
>>
```

Workspace Cont…

- Display contents of workspace

```
>> whos
 Name      Size                Bytes  Class
  a         1x1                    8  double array
  b         1x1                    8  double array
  c         1x1                    8  double array
Grand total is 3 elements using 24 bytes
>>
```

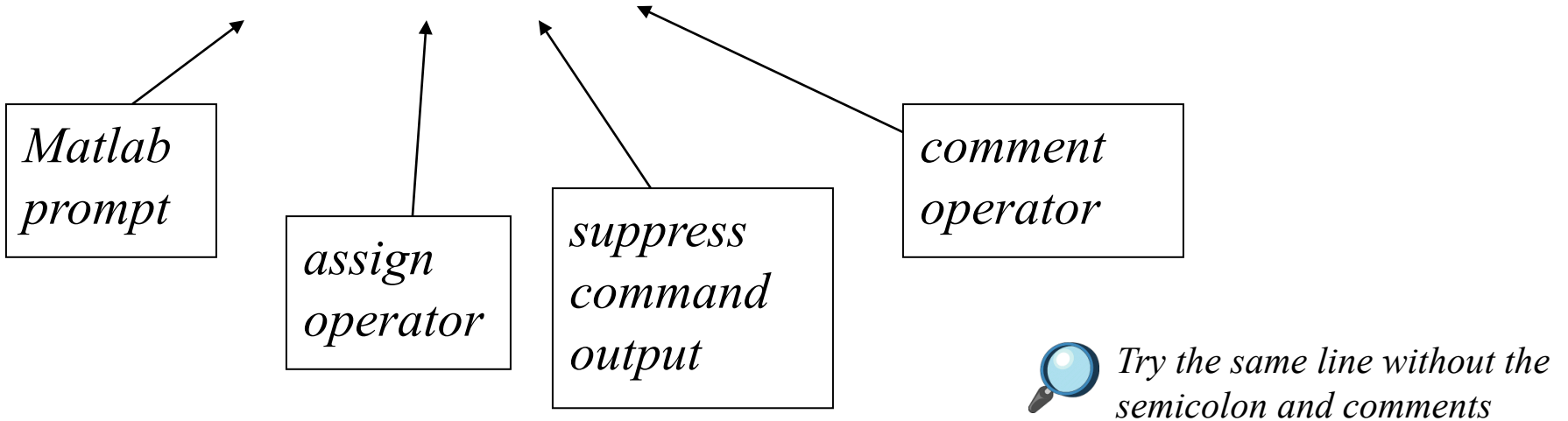
- Delete variable(s) from workspace

```
>> clear a b; % delete a and b from workspace
>> whos
>> clear all; % delete all variables from workspace
>> whos
```

Variables

- Don’t have to declare type
- Don’t even have to initialise
- Just assign in command window

```
>>
>> a=12; % variable a is assigned 12
```



Try the same line without the semicolon and comments

Variables Cont...

- Variable names:
  - Must start with a letter
  - May contain only letters, digits, and the underscore "_"
  - Matlab is case sensitive, i.e. one & OnE are different variables.

- Assignment statement:
  - *Variable = number;*
  - *Variable = expression;*

- Example:

```
>> tutorial = 1234;
>> tutorial = 1234
tutorial =
      1234
```

NOTE: when a semi-colon ";" is placed at the end of each command, the result is not displayed.

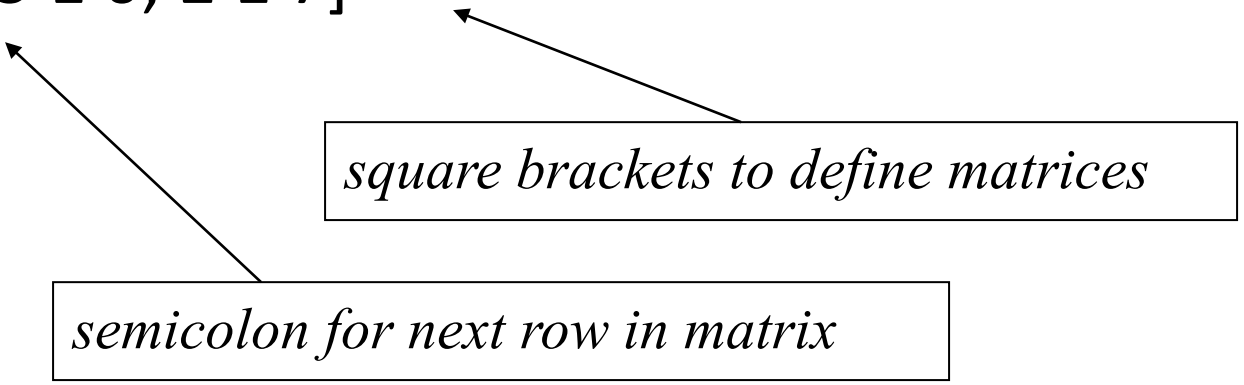
Variables Cont...

- Special variables:
  - ans : default variable name for the result
  - pi: π = 3.1415926……
  - eps: $\in$ = 2.2204e-016, smallest amount by which 2 numbers can differ.
  - Inf or inf : ∞, infinity
  - NaN or nan: not-a-number

Matrices

- Don’t need to initialise type, or dimensions

```
>>A = [3 2 1; 5 1 0; 2 1 7]
A =
     3     2     1
     5     1     0
     2     1     7
>>
```



Manipulating Matrices

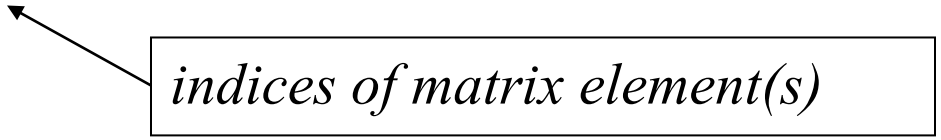
$$A = \begin{matrix} 3 & 2 & 1 \\ 5 & 1 & 0 \\ 2 & 1 & 7 \end{matrix}$$

- Access elements of a matrix

>>A(1,2)

ans=

2



- Remember Matrix(row,column)
- Naming convention Matrix variables start with a capital letter while vectors or scalar variables start with a simple letter

The : Operator

- VERY important operator in Matlab
- Means ‘to’

```
>> 1:10
ans =
    1    2    3    4    5    6    7    8    9   10
>> 1:2:10
ans =
    1    3    5    7    9
```

Try the following

```
>> x=0:pi/12:2*pi;
>> y=sin(x)
```

The : Operator and Matrices

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

$$A = \begin{matrix} 3 & 2 & 1 \\ 5 & 1 & 0 \\ 2 & 1 & 7 \end{matrix}$$

🔎 *What'll happen if you type A(:,:) ?*

Manipulating Matrices

```
>> A '      % transpose
>> B*A      % matrix multiplication
>> B.*A     % element by element multiplication
>> B/A      % matrix division
>> B./A     % element by element division
>> [B A]    % Join matrices (horizontally)
>> [B; A]   % Join matrices (vertically)
```

$$A = \begin{matrix} 3 & 2 & 1 \\ 5 & 1 & 0 \\ 2 & 1 & 7 \end{matrix}$$

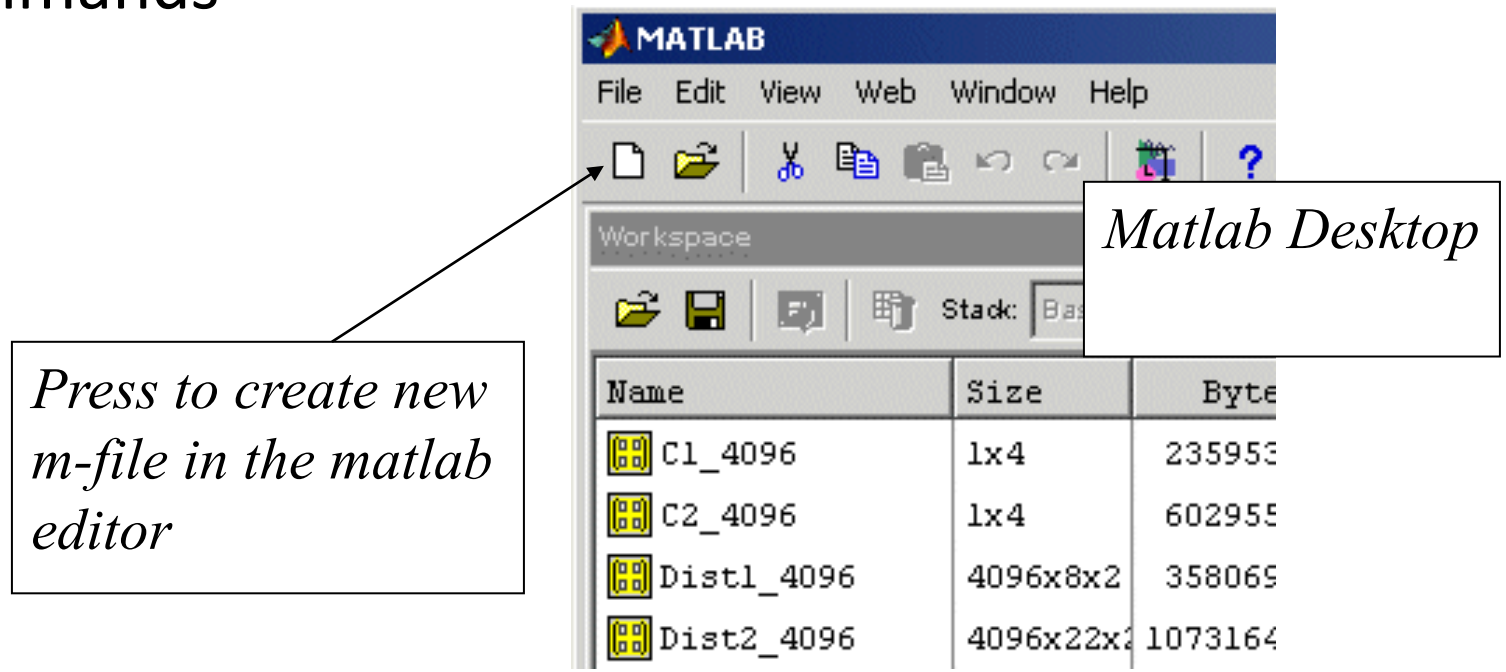
$$B = \begin{matrix} 1 & 3 & 1 \\ 4 & 9 & 5 \\ 2 & 7 & 2 \end{matrix}$$

Enter matrix B into the Matlab workspace

Create matrices A and B and try out the the matrix operators in this slide

Scripts

- Matlab editor
- Use scripts to execute a series of Matlab commands



Scripts

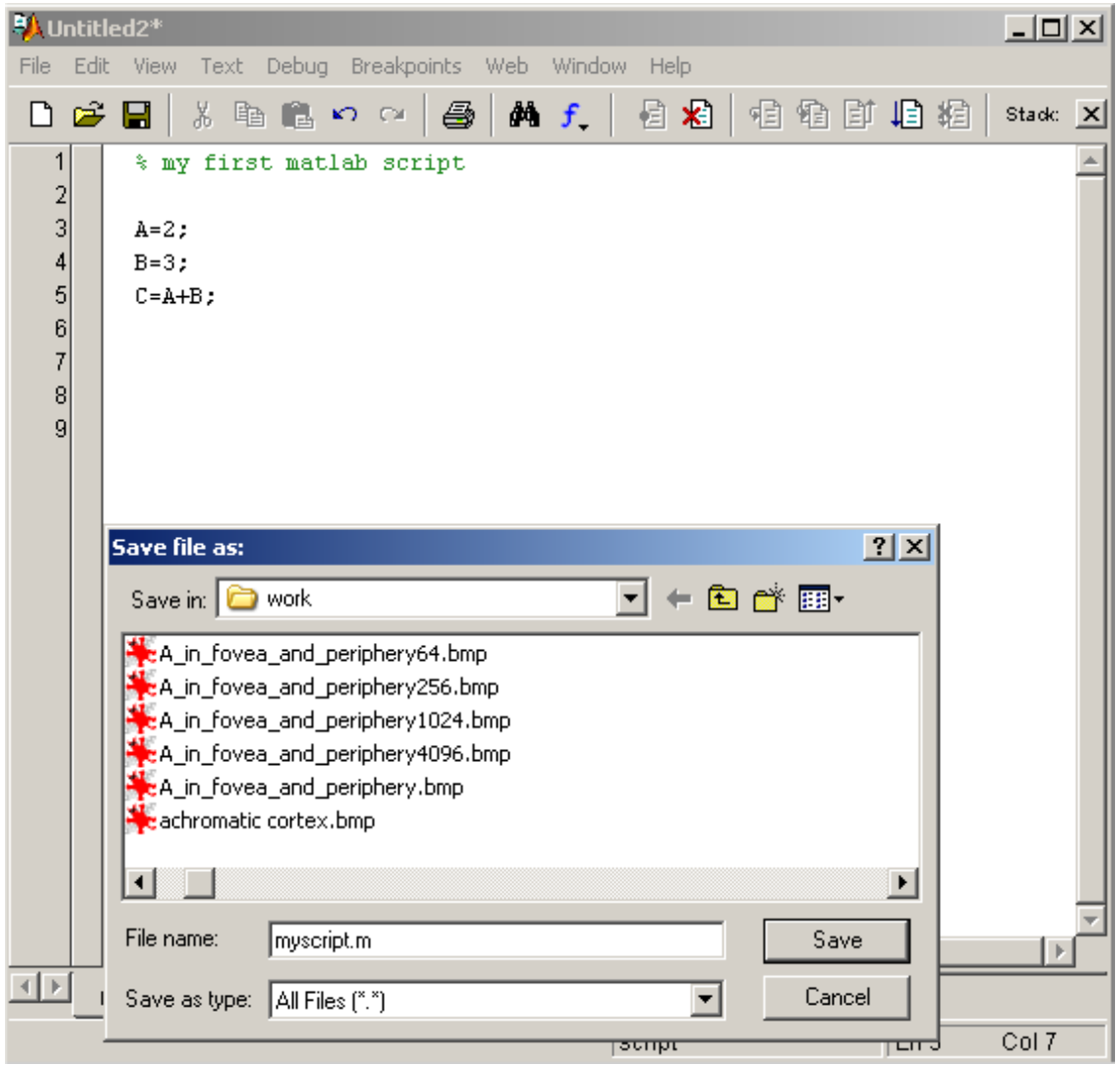
- Scripts will manipulate and store variables and matrices in the Matlab Workspace (memory).
- They can be called from the Matlab command line by typing the (case sensitive!) filename of the script file.

  >> myscript

- Scripts can be opened in the editor by the following

  >> open myscript

  Highlight a few lines of your script by left- clicking and dragging the mouse over the lines. Right-click the highlighted lines and select Evaluate Selection.

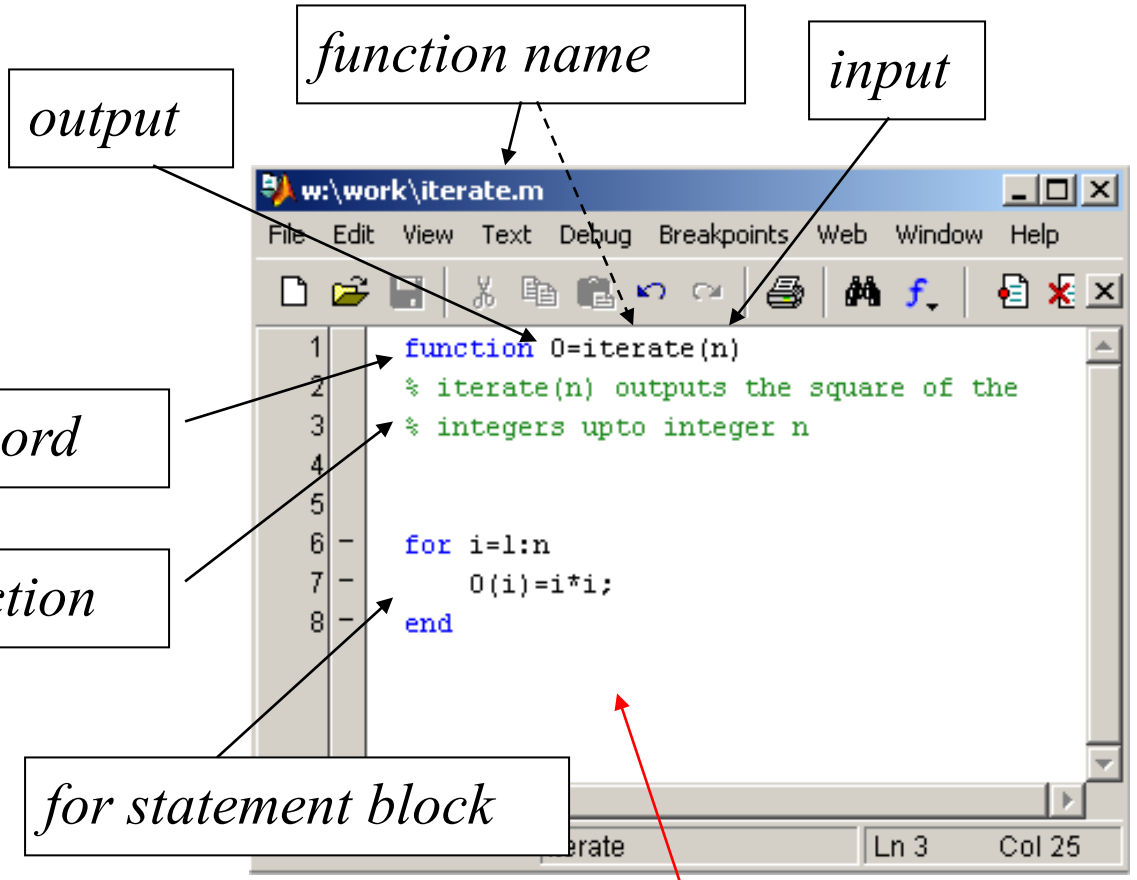


Functions

- Programming in Matlab.
- Users can write functions which can be called from the command line.
- Functions can accept input variable(s)/matrice(s) and will output variable(s)/matrice(s).
- Functions will **not** manipulate variable(s)/matrice(s) in the Matlab Workspace.
- In Matlab functions closely resemble scripts and can be written in the Matlab editor. Matlab functions have the function keyword.
- Remember that the filename of a function will be its calling function name.
- Don’t overload any built-in functions by using the same filename for your functions or scripts!
- Functions can be opened for editing using the **open** command. Many built-in Matlab functions can also be viewed using this command.

Functions Cont…

```
>> I=iterate(5)
I =
     1     4     9    16    25
```



Access the comments of your Matlab functions
>> help iterate

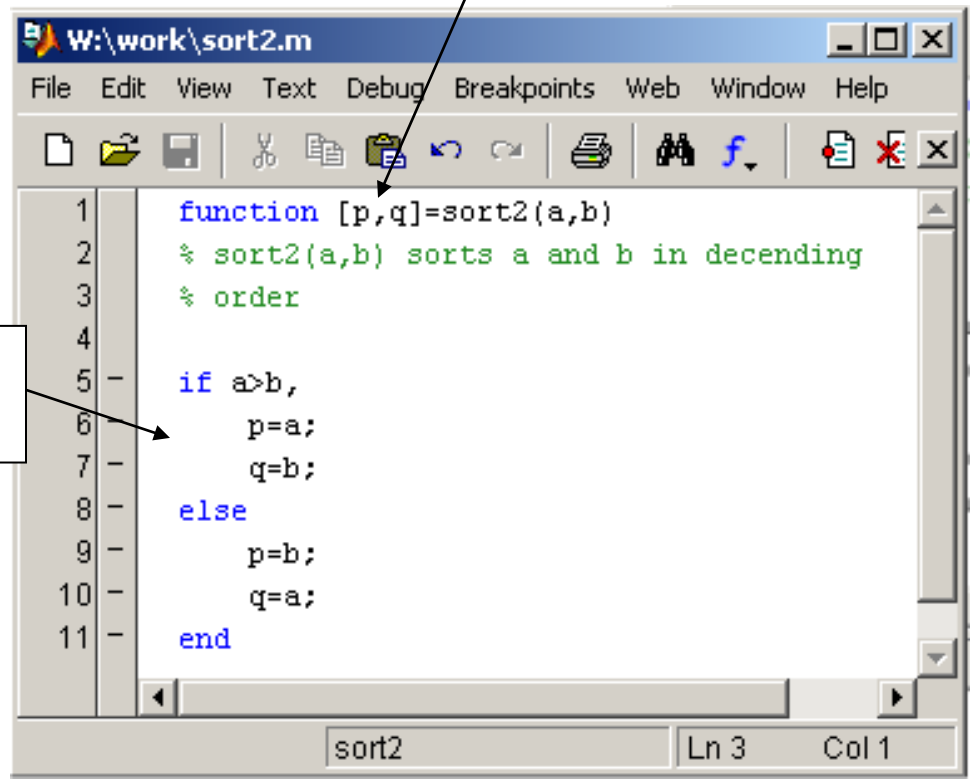
Make sure you save changes to the m-file before you call the function!

Functions Cont…

```
>> [i j]=sort2(2,4)
i =
     4
j =
     2
>>
```

Functions can have many outputs contained in a matrix

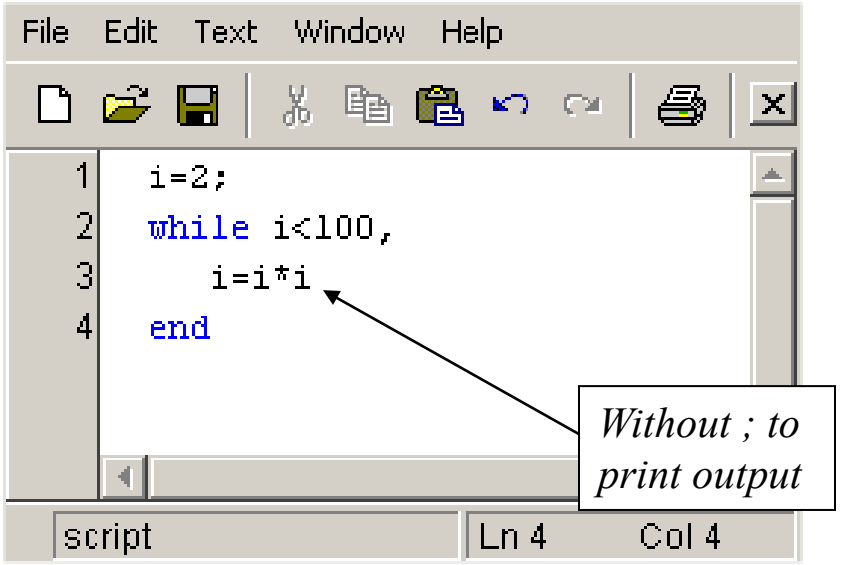
if statement block



Remember to use the Matlab help command for syntax
>> help if

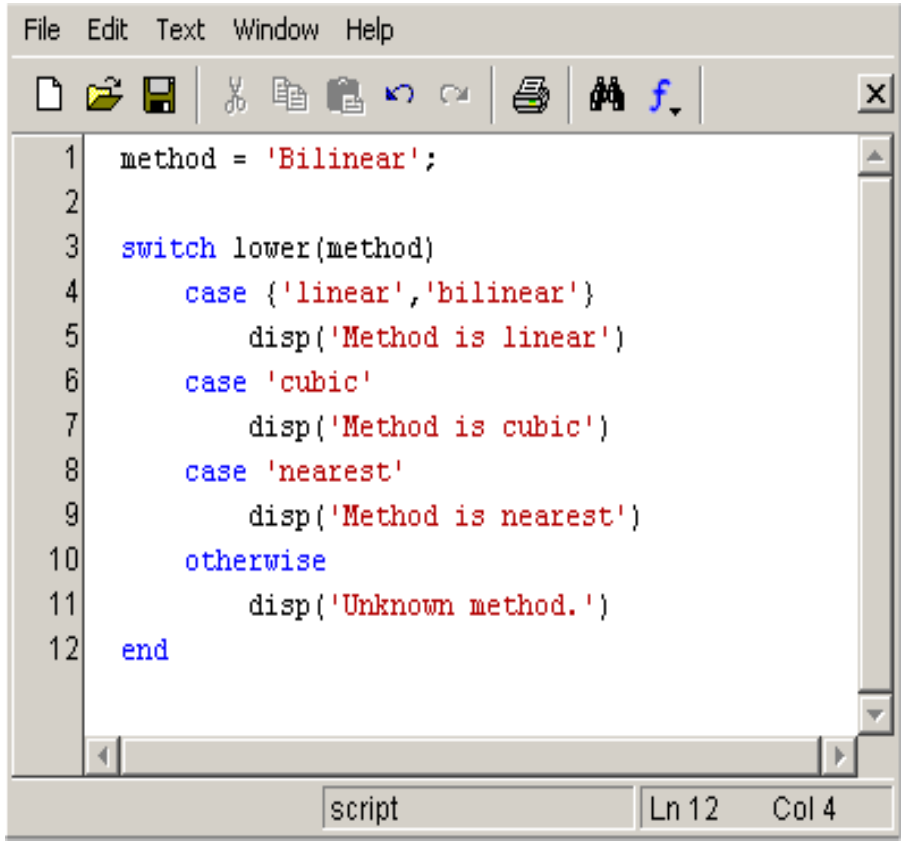
Flow Control

While statement block




```
i =
     4
i =
    16
i =
   256
```

Switch statement block




```
Method is linear
>>
```

Debugging

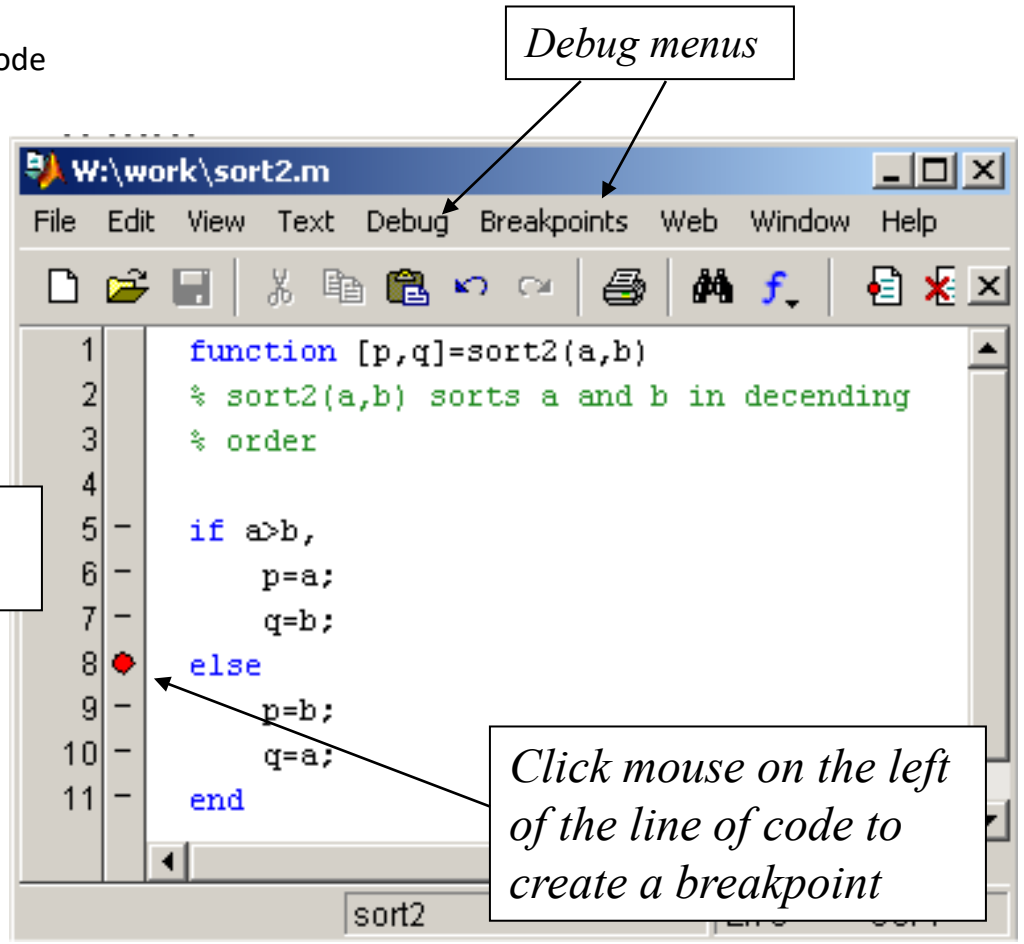
- Set breakpoints to stop the execution of code

```
>> [i j]=sort2(2,4)
K>>
K>> whos
 Name      Size              Bytes  Class
  a         1x1                 8  double array
  b         1x1                 8  double array
Grand total is 2 elements using 16 bytes
K>> a
a =
     2
K>> return
i =

     4
j =
     2
```

local function workspace

exit debug mode



Vectors, Matrices and Linear Algebra

- Vectors
- Matrices
- Solutions to Systems of Linear Equations.

Vectors

Example:

```
>> x = [ 0   0.25*pi   0.5*pi   0.75*pi   pi ]
x =
        0    0.7854    1.5708    2.3562    3.1416
>> y = [ 0; 0.25*pi; 0.5*pi; 0.75*pi; pi ]
y =
        0
   0.7854
   1.5708
   2.3562
   3.1416
```

x is a row vector.

y is a column vector.

Vectors Cont…

- Vector Addressing – A vector element is addressed in MATLAB with an integer index enclosed in parentheses.
- Example:

```
>> x(3)
ans =
   1.5708
```

← *3rd element of vector x*

- The colon notation may be used to address a block of elements.

(start : increment : end)

start is the starting index, increment is the amount to add to each successive index, and end is the ending index. A shortened format (start : end) may be used if increment is 1.

- Example:

```
>> x(1:3)
ans =
        0    0.7854    1.5708
```

← *1st to 3rd elements of vector x*

Vectors Cont…

Some useful commands:

x = start:end	create row vector x starting with start, counting by one, ending at end
x = start:increment:end	create row vector x starting with start, counting by increment, ending at or before end
linspace(start,end,number)	create row vector x starting with start, ending at end, having number elements
length(x)	returns the length of vector x
y = x’	transpose of vector x
dot (x, y)	returns the scalar dot product of the vector x and y.

Matrices

- *A Matrix array is two-dimensional, having both multiple rows and multiple columns, similar to vector arrays:*
  - *it begins with [, and end with]*
  - *spaces or commas are used to separate elements in a row*
  - *semicolon or enter is used to separate rows.*

A is an m x n matrix.

$$A = \begin{bmatrix} a_{11} & a_{12} & a_{13} & \cdots & a_{1n} \\ a_{21} & a_{22} & a_{23} & \cdots & a_{2n} \\ a_{31} & a_{32} & a_{33} & \cdots & a_{3n} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ a_{m1} & a_{m2} & a_{m3} & \cdots & a_{mn} \end{bmatrix}$$

the main diagonal

- *Example:*
- *>> f = [1 2 3; 4 5 6]*

$$f = \begin{matrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{matrix}$$

Matrices Cont…

- Matrix Addressing:
  -- *matrixname(row, column)*
  -- **colon** may be used in place of a row or column reference to select the entire row or column.

  - *Example:*

```
>> f(2,3)

ans =

     6

>> h(:,1)

ans =

     2

     1
```

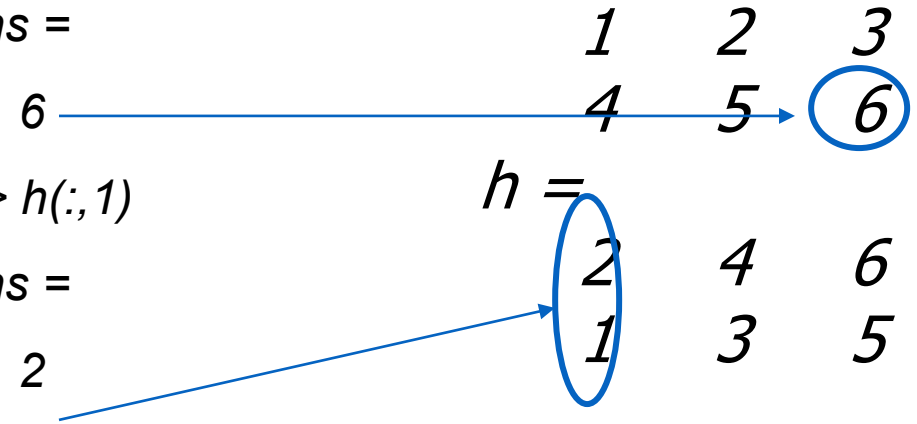
recall:

```
f =

     1     2     3
     4     5     6

h =

     2     4     6
     1     3     5
```



Matrices Cont…

more commands

Transpose	B = A′
Identity Matrix	eye(n) ➔ returns an n x n identity matrix eye(m,n) ➔ returns an m x n matrix with ones on the main diagonal and zeros elsewhere.
Addition and subtraction	C = A + B C = A – B
Scalar Multiplication	B = αA, where α is a scalar.
Matrix Multiplication	C = A*B
Matrix Inverse	B = inv(A), A must be a square matrix in this case. rank (A) ➔ returns the rank of the matrix A.
Matrix Powers	B = A.^2 ➔ squares each element in the matrix C = A * A ➔ computes A*A, and A must be a square matrix.
Determinant	det (A), and A must be a square matrix.

A, B, C are matrices, and m, n, α are scalars.

Solutions to Systems of Linear Equations

- Example: a system of 3 linear equations with 3 unknowns (x_1, x_2, x_3):

$$\begin{aligned} 3x_1 + 2x_2 - x_3 &= 10 \\ -x_1 + 3x_2 + 2x_3 &= 5 \\ x_1 - x_2 - x_3 &= -1 \end{aligned}$$

Let :

$$A = \begin{bmatrix} 3 & 2 & 1 \\ -1 & 3 & 2 \\ 1 & -1 & -1 \end{bmatrix} \qquad x = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} \qquad b = \begin{bmatrix} 10 \\ 5 \\ -1 \end{bmatrix}$$

Then, the system can be described as:

$$Ax = b$$

Solutions to Systems of Linear Equations (con't…)

- Solution by Matrix Inverse:

  $Ax = b$

  $A^{-1}Ax = A^{-1}b$

  $x = A^{-1}b$

- MATLAB:

```
>> A = [ 3 2 -1; -1 3 2; 1 -1 -1];
>> b = [ 10; 5; -1];
>> x = inv(A)*b
x =
   -2.0000
    5.0000
   -6.0000
```

Answer:
$x_1 = -2,\ x_2 = 5,\ x_3 = -6$

- Solution by Matrix Division:

  The solution to the equation

  $$Ax = b$$

  can be computed using left division.

- *MATLAB:*

```
>> A = [ 3 2 -1; -1 3 2; 1 -1 -1];
>> b = [ 10; 5; -1];
>> x = A\b
x =
   -2.0000
    5.0000
   -6.0000
```

Answer:
$x_1 = -2,\ x_2 = 5,\ x_3 = -6$

NOTE:

left division: $A\backslash b$ ➔ $b \div A$ right division: x/y ➔ $x \div y$

Flow Control: if…else

Example: (**if**…**else** and **elseif** clauses)

```
if temperature > 100
    disp (‘Too hot – equipment malfunctioning.’)
elseif temperature > 90
    disp (‘Normal operating range.’);
else
    disp (‘Too cold – turn off equipment.’)
end
```

Flow Control: loops

- **for** loop

  for *variable = expression*

  commands

  end

- **while** loop

  while *expression*

  commands

  end

- *Example (for loop):*

```
for t = 1:5000
    y(t) = sin (2*pi*t/10);
end
```

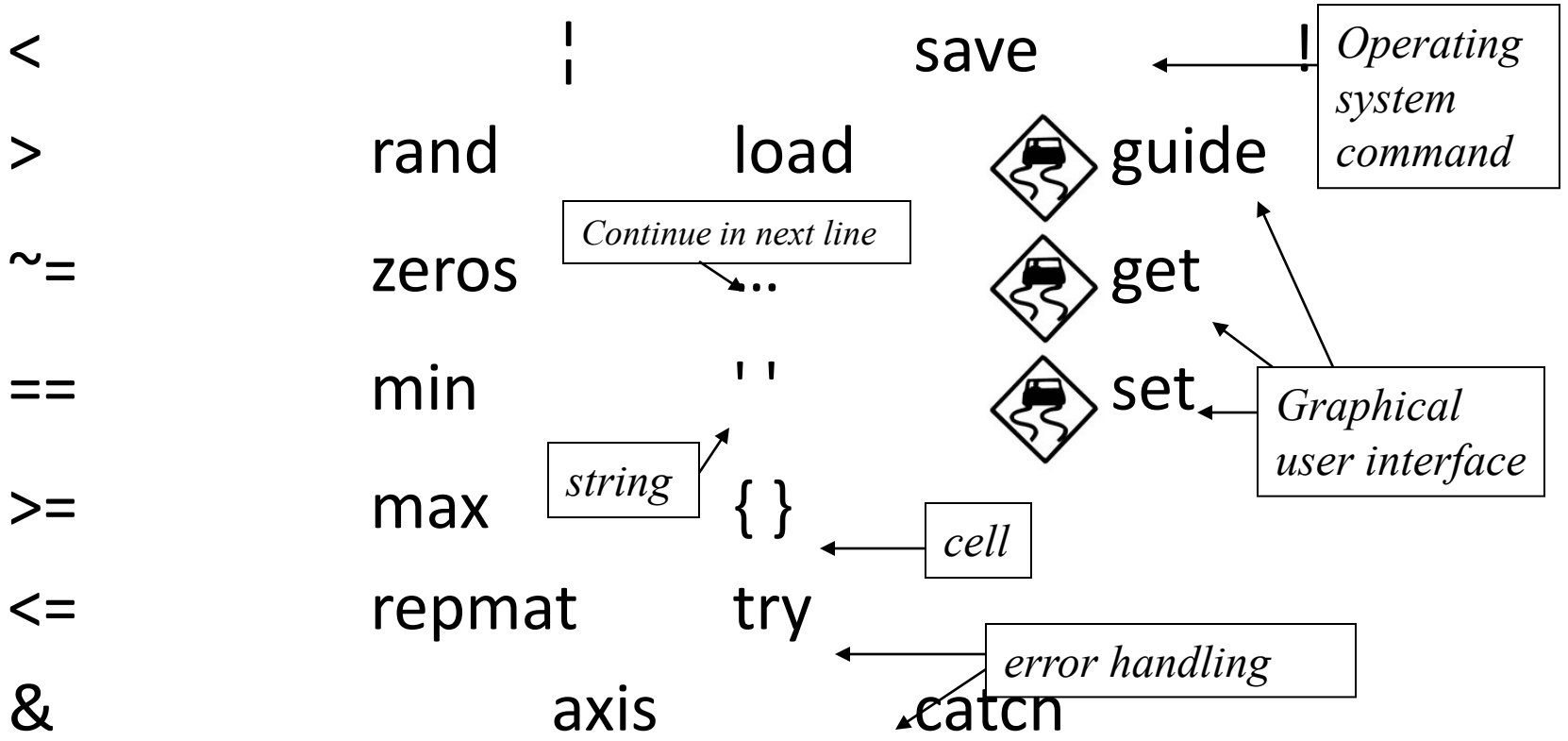
- *Example (while loop):*

```
EPS = 1;
while ( 1+EPS) >1
    EPS = EPS/2;
end
EPS = 2*EPS
```

- *the **break** statement*

  ***break** – is used to terminate the execution of the loop.*

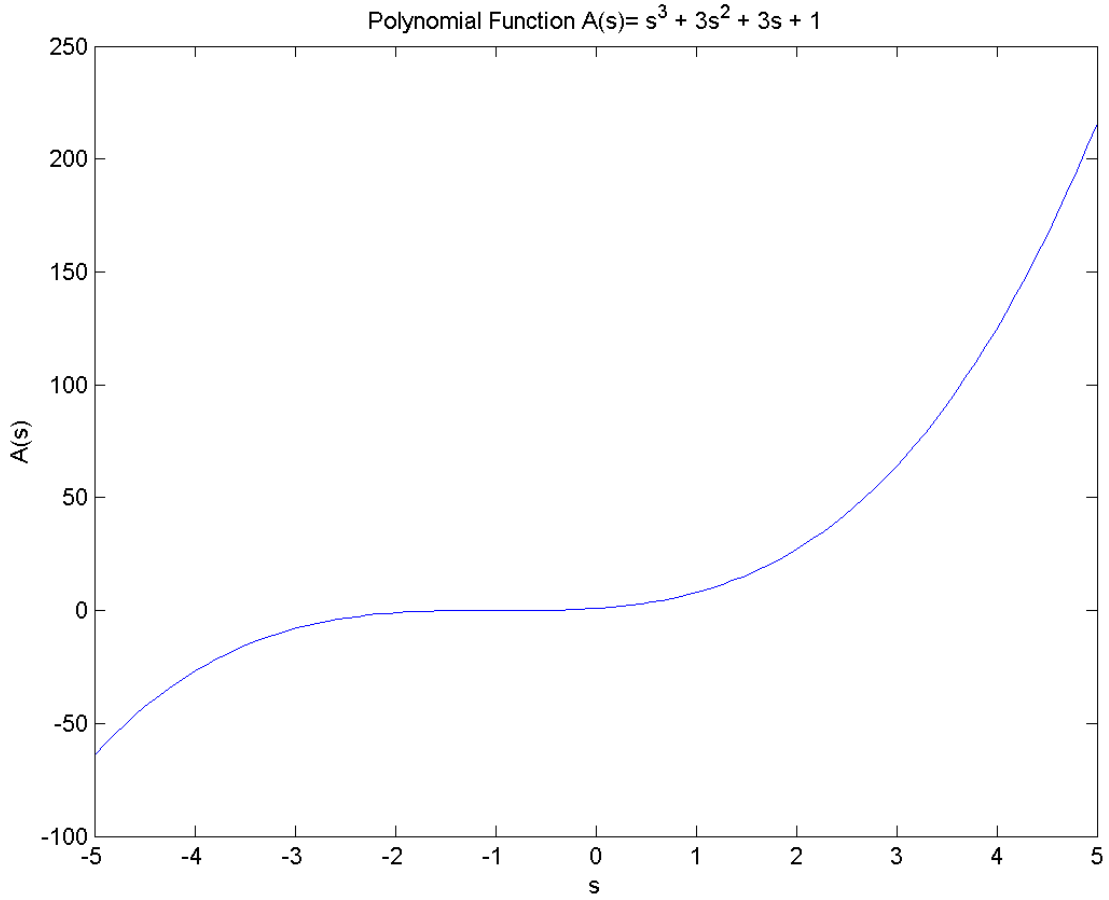
Useful operators and built-in functions



Remember to use the Matlab help command if you get stuck

2 Part II: Visualization

Visualization: Plotting



- Example:

```
>> s = linspace (-5, 5, 100);
>> coeff = [ 1 3 3 1];
>> A = polyval (coeff, s);
>> plot (s, A),
>> xlabel ('s')
>> ylabel ('A(s)')
```

Plotting (con’t)

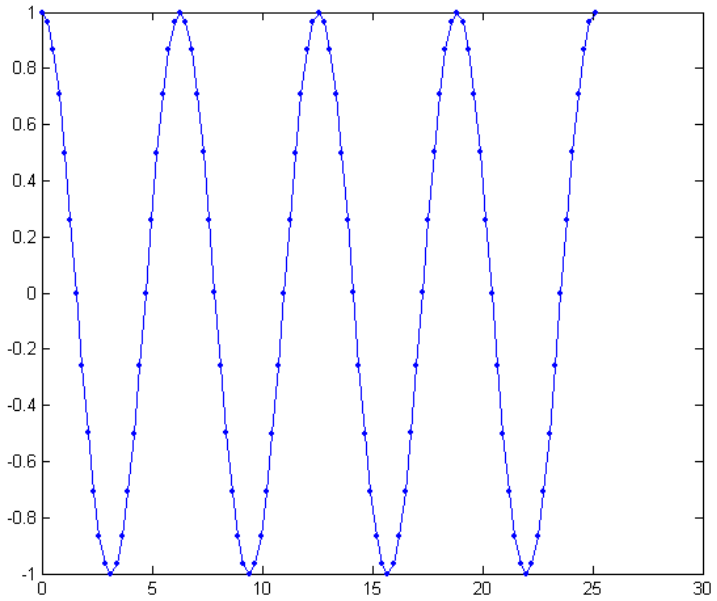
- *Plot a Helix*

```
t = linspace (-5, 5, 101);
x = cos(t);
y = sin(t);
z = t;
plot3(x,y,z);
box on;
```

Visualization - plotting data

```
>> figure   % create new figure
>> t=0:pi/12:8*pi;
>> y=cos(t);
>> plot(t,y,'b.-')
```

Plot style



A = amplitude
phi = phase
*w = angular frequency = 2*pi*frequency*

Investigate the function
*>> y=A*cos(w*t+phi);*
for different values of phi (eg: 0, pi/4, pi/3, pi/2), w (eg: 1, 2, 3, 4) and A (eg: 1, 0.5, 2).
*Use the **hold on** Matlab command to display your plots in the same figure. Remember to type **hold off** to go back to normal plotting mode.*
*Try using different plot styles (**help plot**)*

3 Part III: Open Image File

Open and display image

- Imread(‘imagename’);
- Imshow(image);

Open File

- fileID = fopen('dokument1.txt','r');
- C = textscan(fileID,'%c'); %bisa juga %s
- fclose(fileID);

Lifelong Learning

THANKS YOU

Sources

Introduction to Matlab, Sumitha Balasuriya