MATLAB Basic

1. System commands

```
save <filename>                 load <filename>

who       % show var            whos      % var details
clear /x?/ % del x or all

input(s)  % show s and wait for input
format    % switch short(4) / long(16) / bank(2) /
          % short e / long e / rat / + (pos, neg, 0) /
          % compact / loose

disp      % display value without "var = "
fscanf    % Read formatted data from a file
fprintf   % Formatted writes to screen or file
          % MUST have format char
%s        % string              %d        % integer
%f        % float               %e        % %f + exp
%g        % whichever more compact in %f or %e
```

2. Notations

```
%         % comments            ... % continue line
;         % hide output
```

3. Var

a. Declare and assign
b. `x = 3` → 1 X 1 matrix
c. Multi-assign: `a = 2; b = 7; c = a * b`
d. Vector: 1-d array
   i. Row vector, space/comma delimit
   ii. Col vector, ; delimit

```
r = [7 8 9 10 11]; t = [2, 3, 4, 5, 6]; res = r + t
% res = 9  11  13  15 17          % row vector

c = [7;  8;  9;  10; 11]
% res = 7\n 8\n 9\n 10\n 11       % col vector
```

e. Special variables

```
Eps                     % Smallest incremental number
```

4. Vector, Matrix and Array Commands

```
m = [1, 2, 3; 4 5 6; 7, 8, 9]
% m = 1 2 3\n 4 5 6\n 7 8 9\n
```

5. Data structures

a. `string` consists of `char`.
b. `function handle`: pointer

```
str = 'Hello!',         n = 2345,          d = double(n)
un = uint32(789.50),rn = 5678.92347,       c = int32(rn)
logical(10) = 1         logical(0) = 0
```

c. determine ds: `is...`

6. Operators

a. Assignment: NO `++` and `+=`
b. Arithmatic: A, B

```
+, -, .*, ./, .\        % A, B sais me size or one scalar
*                       % dot product, a.col = b.row
.*                      % e-wise prod, same size
/                       % B* inv(A)
~=                      % not eq
```

c. Logical

```
~         % not, highest precedence
&         % and, equal precedence with or
|         % or
```

7. Condition

```
if ...;  elseif...; else...; end
switch(var); case <>; ...; otherwise; ...; end
```

8. Loop: break, continue

```
while n > 0; ...; n = n – 1; end;

for a = start:step:end;...      % default step is 1
                                % both end inclusive
for a = Array; ...
```

MATLAB Vector

1. Intro

```
r = [1, 2, 3, 4]        % row vector
c = [1; 2; 3; 4]        % col  vector
v = [first:step:end]    % vector

r(start:end)            % access, BOTH END INCLUSIVE
```

2. Ops

a. Same type, same size or one is scalar

```
r1 = [1, 2, 3];         % row
c1 = [1; 2; 3];         % col vector
s1 = [1];               % scalar

disp(r1 + r1)           % 2, 4, 6
disp(c1 + c1)           % 2\n 4\n 6
disp(r1 + s1)           % 2, 3, 4
% same goes to -, .*, ./, /\
```

b. Scalar Multiplication

```
r1 = [1, 2, 3];         % row
s1 = [5];

disp(r1 * s1)
```

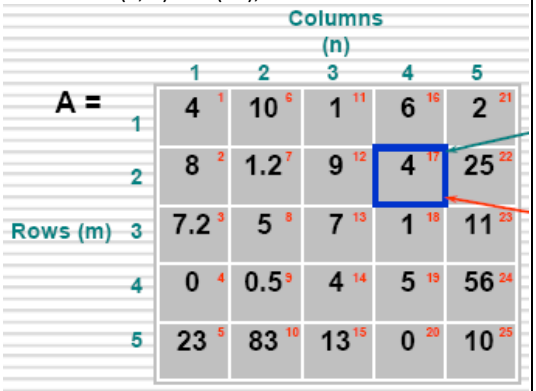
c. ': transpose, row <=> col
d. Appending: horizontal: `[r1, r2]`; vertical: `[r1; r2]`
e. Magnitude: |v| = sqrt (sigma(v_i^2))

```
|v| = sqrt(sum(v.*v))          % element wise product
```

f. Dot product: `dot(vec1, vec2)`

MATLAB Matrices

1. All variables are matirices ⇔ array.
   a. Vectors: matrices w/ 1 row OR 1 col
   b. Scalar: matrices w/ 1 row AND 1 col
2. Access: A(2, 4) or A(17), **START FROM 1**.



3. Extension

```
B = [1, 2; 3, 4]        % repeate matix in each dimension
Br = repmat(B, 1, 2)    % reapmat(M, d1, ..., dn)

Bc = [B, 2*B; 3*B, B^2]          % concatenate
```

4. Functions

```
zeros(n)        % n * n array of 0s
ones(n)         % n * n array of 1s
eye(n)          % n * n unit matrix
rand(n)         % uniformly dist rand (0, 1)
magic(n)        % row, col, diag same-sum array

inv(M)          % inverse
```

5.

MATLAB Array

1. Vecter & matrices are 1d/2d special case of arrays
2. Some special arrays
3. Multi dimensional array
   a. Create

```
A = [1 2 3; 4 5 6; 7 8 9]          % 2-d array
A(:, :, 2) = [0 0 0; 0 0 0; 0 0 0]  % add a dimension

B = cat(d, A1, ..., An)  % concatenate along dimension d
B = [A1, A2...]
```

b. Colon notation
   i. Create vector: `start:step:end`
   ii. Slice array: `A(start: end)` similar to Python `[start: end:step]` notation, BUT both end included, **INDEX start from 1.**

```
A = 1:1:10              % 1..10
A = linspace(start, end, num)
```

c. Special multi-D arrays

```
zeros(n_1, ..., n_n)    % multi-D array of 0s
ones(n_1, ..., n_n)     % multi-D array of 1s
rand(n_1, ..., n_n)     % multi-D rand array
magic(n_1, ..., n_n)    % multi-D magic array
```

4. Array Functions

```
length(a)               % largest length in all dimensions
ndims                   % number of dimensions
numel                   % number of elements
size                    % length of each dimension

sort(a, n)              % sort along n-th dimcell

iscolumn                % determin col vector
isrow                   % determin row vector
ismatrix                % determin matrx
isempty                 % determin empty
is...                   % true / false

reshape                 % reshape to another dimension
```

MATLAB Functions

1. Any fuction EXCEPT anonymous one must be in file `<name.m>`. 1 + n sub-functions.
   a. Primary function: only 1, at first, callable from outside.
   b. Sub-functions: any number, comes after. Only visible in the file scope
   c. Private funcitons: in `private` folder, only visible to functions in the parent folder

```
function [out1, o2...] = name(var1, var2...)
% instructions, help(func_name) to show
end
```

2. Anonymous functions

```
Name = @(args)expression

power = @(x, n) x.^n
```

3. Glocal variable: declare before use

```
Global TOTAL
```

MATLAB I/O

1. High level read API

```
% read from file
A = importdata(filename,delimiterIn,headerlinesIn)
% read from clipboard
A = importdata('-pastespecial')

for k = [1:7]                   % view data
    disp(A.colheaders{1, k})
    disp(A.data(:, k))
    disp(' ')
end
```

2.

MATLAB Plotting

1.

```
hold on                 % preserve old graphic
hold off                % clear old graphic
```

2.

MATLAB OOP

1. Structure

```
classdef name<superclass
    properties                  # instance vars
        var;
    end

    methods                     # instance method
        function self = set(args)
            self.var = args;
        end
    end

    methods (Static = true)     # class method
        function classMethd(agrs)
            fprintf("hello %s", agrs)
        end
    end
end
```

2. An example

```
classdef Dog
    properties      % instance var
        name;
        age;
        msg;
    end

    methods         % instance method
        function obj = Dog()                % constructor
        end

        function obj = setInfo(obj, name, age)
            obj.name = name;
            obj.age = age;
        end

        function rst = bark(obj, times)
            rst = '';
            for i = 1:times;
                rst = ['Hello, dog ', obj.name, '! ', rst];
            end
        end
    end
end
```

3. Vecter & matrices