APPLICATIONS OF MATLAB IN ENGINEERING

Yan-Fu Kuo
Dept. of Bio-industrial Mechatronics Engineering
National Taiwan University

Fall 2015

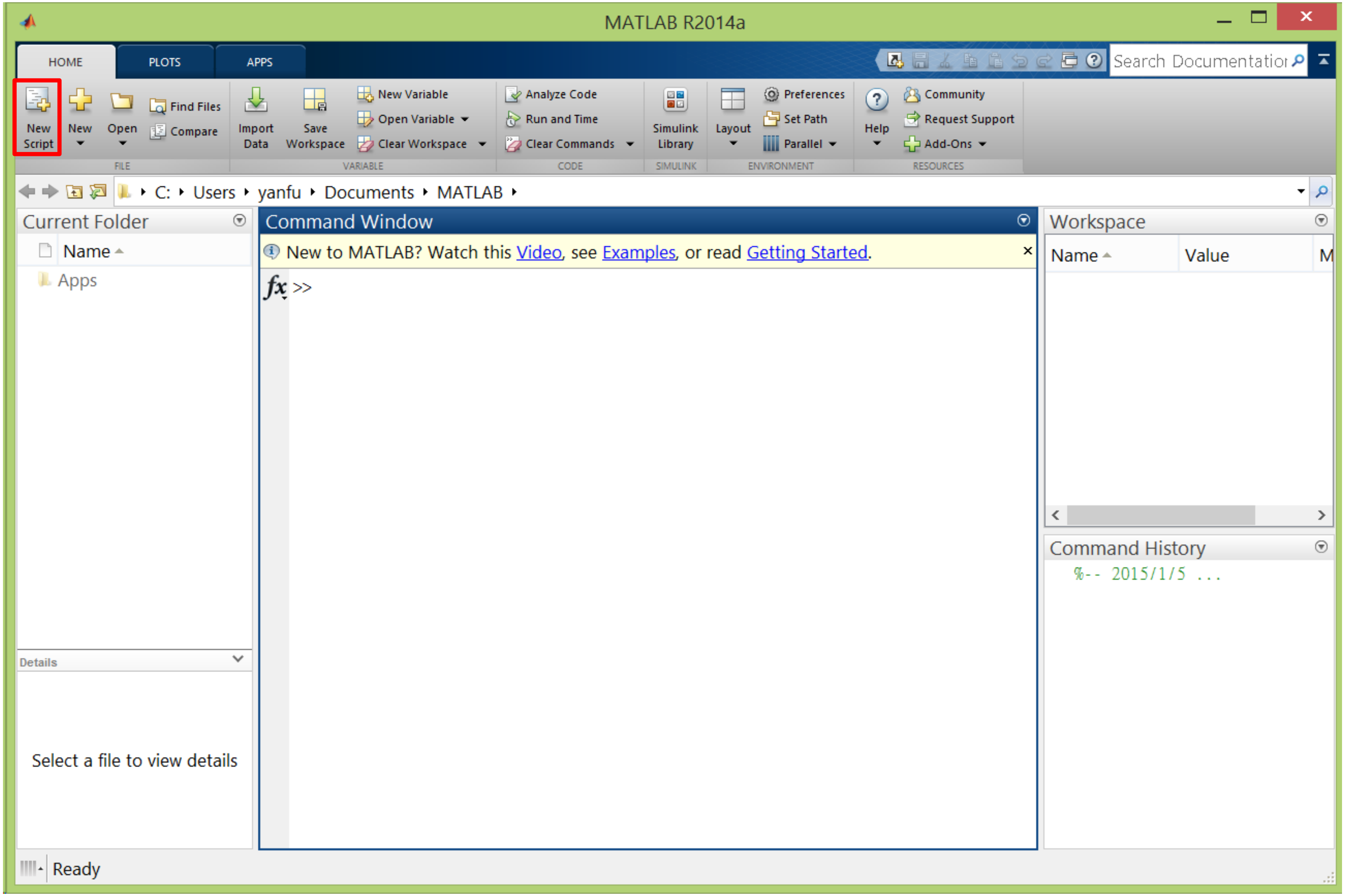
Today:

- Script writing
- Structured programming
- User-defined function

MATLAB Script

- A file containing a series of MATLAB commands
- Pretty much like a C/C++ program
- Scripts need to be saved to a `<file>.m` file before they can be run

Start A Script (.m) File



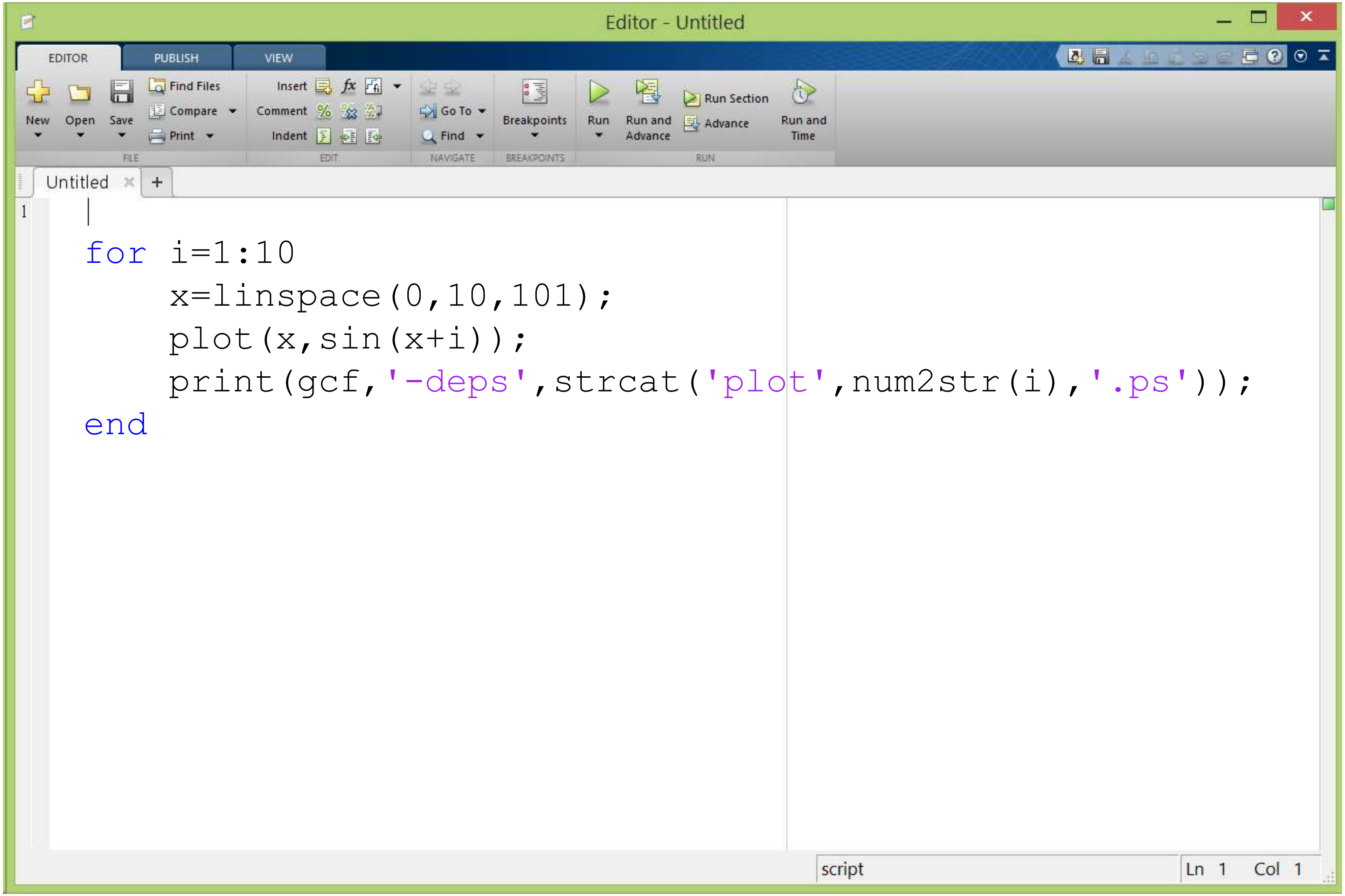
Select a file to view details

New to MATLAB? Watch this Video, see Examples, or read Getting Started.

```
fx >>
```

```
%-- 2015/1/5 ...
```

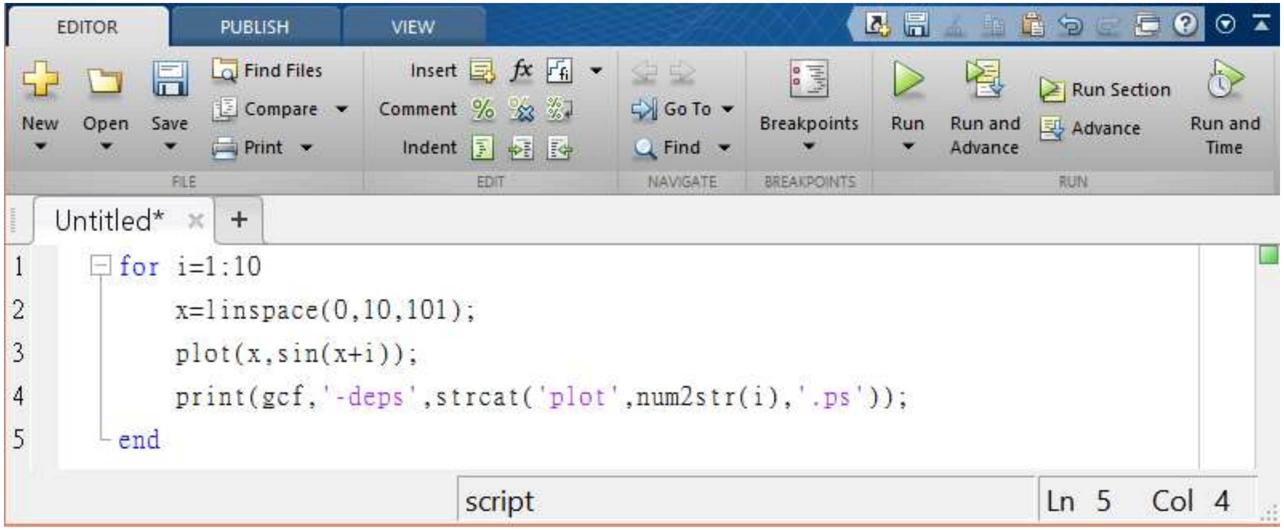
Script Editor



```
for i=1:10
    x=linspace(0,10,101);
    plot(x,sin(x+i));
    print(gcf,'-deps',strcat('plot',num2str(i),'.ps'));
end
```

Script Flow

- Typically scripts run from the first line to the last



```
for i=1:10
    x=linspace(0,10,101);
    plot(x,sin(x+i));
    print(gcf,'-deps',strcat('plot',num2str(i),'.ps'));
end
```

- Structured programming techniques (subroutine, loop, condition, etc) are applied to make the program looks neat

Flow Control

`if, elseif, else`	Execute statements if condition is true
`for`	Execute statements specified number of times
`switch, case, otherwise`	Execute one of several groups of statements
`try, catch`	Execute statements and catch resulting errors
`while`	Repeat execution of statements while condition is true

`break`	Terminate execution of for or while loop
`continue`	Pass control to next iteration of for or while loop
`end`	Terminate block of code, or indicate last array index
`pause`	Halt execution temporarily
`return`	Return control to invoking function

Relational (Logical) Operators

Operator	Meaning
<	Less than
<=	Less than or equal to
>	Greater than
>=	Greater than or equal to
==	Equal to
~=	Not equal to
&&	And
||	Or

if elseif else

```
if condition1
        statement1
elseif condition2
        statement2
else
        statement3
end
```

```
if rem(a, 2) == 0
    disp('a is even')
else
    disp('a is odd')
end
```

- “elseif” and “else” are optional

switch

```
switch expression
case value1
        statement1
case value2
        statement2
.
.
otherwise
        statement
end
```

```matlab
switch input_num
case -1
      disp('negative 1');
case 0
      disp('zero');
case 1
      disp('positive 1');
otherwise
     disp('other value');
end
```

while

```
while expression
          statement
end
```

```
n = 1;
while prod(1:n) < 1e100
     n = n + 1;
end
```

Exercise

- Use while loop to calculate the summation of the series 1+2+3+…+999

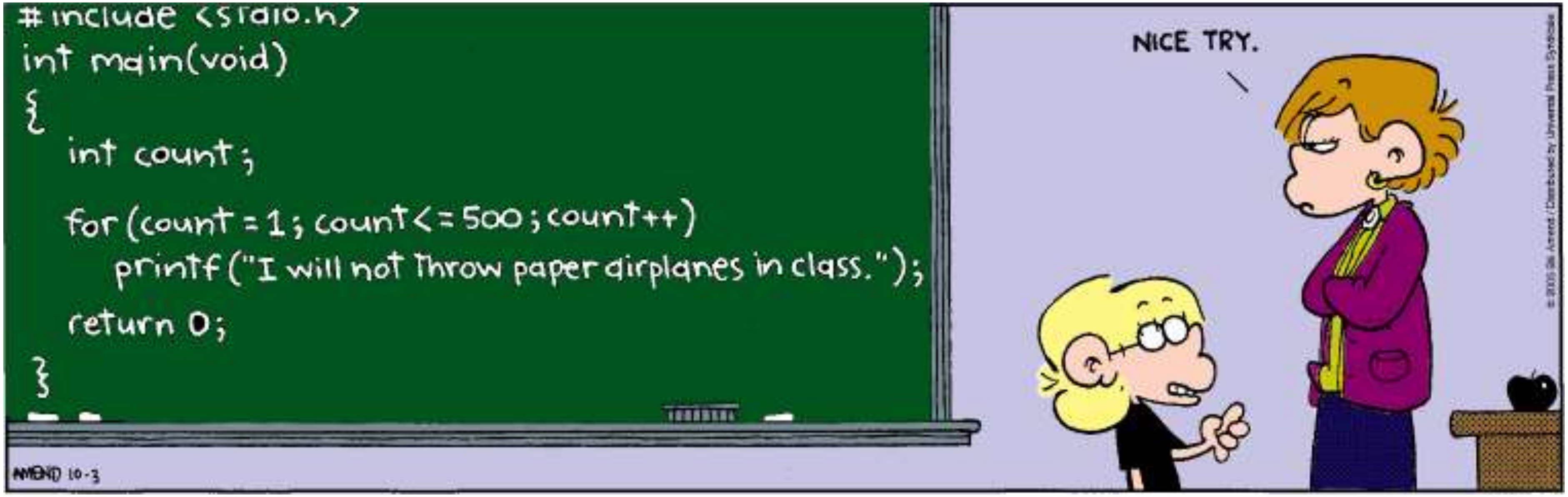
for

for *variable*=*start* : *increment* : *end*

commands

end

```
for n=1:10
        a(n)=2^n;
end
disp(a)
```



```
#include <stdio.h>
int main(void)
{
  int count;

  for(count=1;count<=500;count++)
     printf("I will not Throw paper airplanes in class.");
  return 0;
}
```

Pre-allocating Space to Variables

- In the previous example, we do not pre-allocate space to vector `a` rather than letting MATLAB resize it on every iteration
- Which method is faster?

<table>
<tr><th>A</th><th>B</th></tr>
<tr><td><pre>
tic
for ii = 1:2000
    for jj = 1:2000
        A(ii,jj) = ii + jj;
    end
end
toc
</pre></td><td><pre>
tic
A = zeros(2000, 2000);
for ii = 1:size(A,1)
    for jj = 1:size(A,2)
        A(ii,jj) = ii + jj;
    end
end
toc
</pre></td></tr>
</table>

Exercise

- Use structured programming to:

1. Find the entries in matrix A that are negative
2. Store these entries' position in a matrix B
3. Change the values of these entries to zero

$$A = \begin{bmatrix} 0 & -1 & 4 \\ 9 & -14 & 25 \\ -34 & 49 & 64 \end{bmatrix}$$

break

- Terminates the execution of for or while loops

```
x = 2; k = 0; error = inf;
error_threshold = 1e-32;
while error > error_threshold
    if k > 100
        break
    end
    x = x - sin(x)/cos(x);
    error = abs(x - pi);
    k = k + 1;
end
```

- Used in iteration where convergence is not guaranteed

Tips for Script Writing

- At the beginning of your script, use command
  - `clear all` to remove previous variables
  - `close all` to close all figures
- Use semicolon `;` at the end of commands to inhibit unwanted output
- Use ellipsis `...` to make scripts more readable:

```
A = [1 2 3 4 5 6; ...
     6 5 4 3 2 1];
```

- Press `Ctrl+C` to terminate the script before conclusion

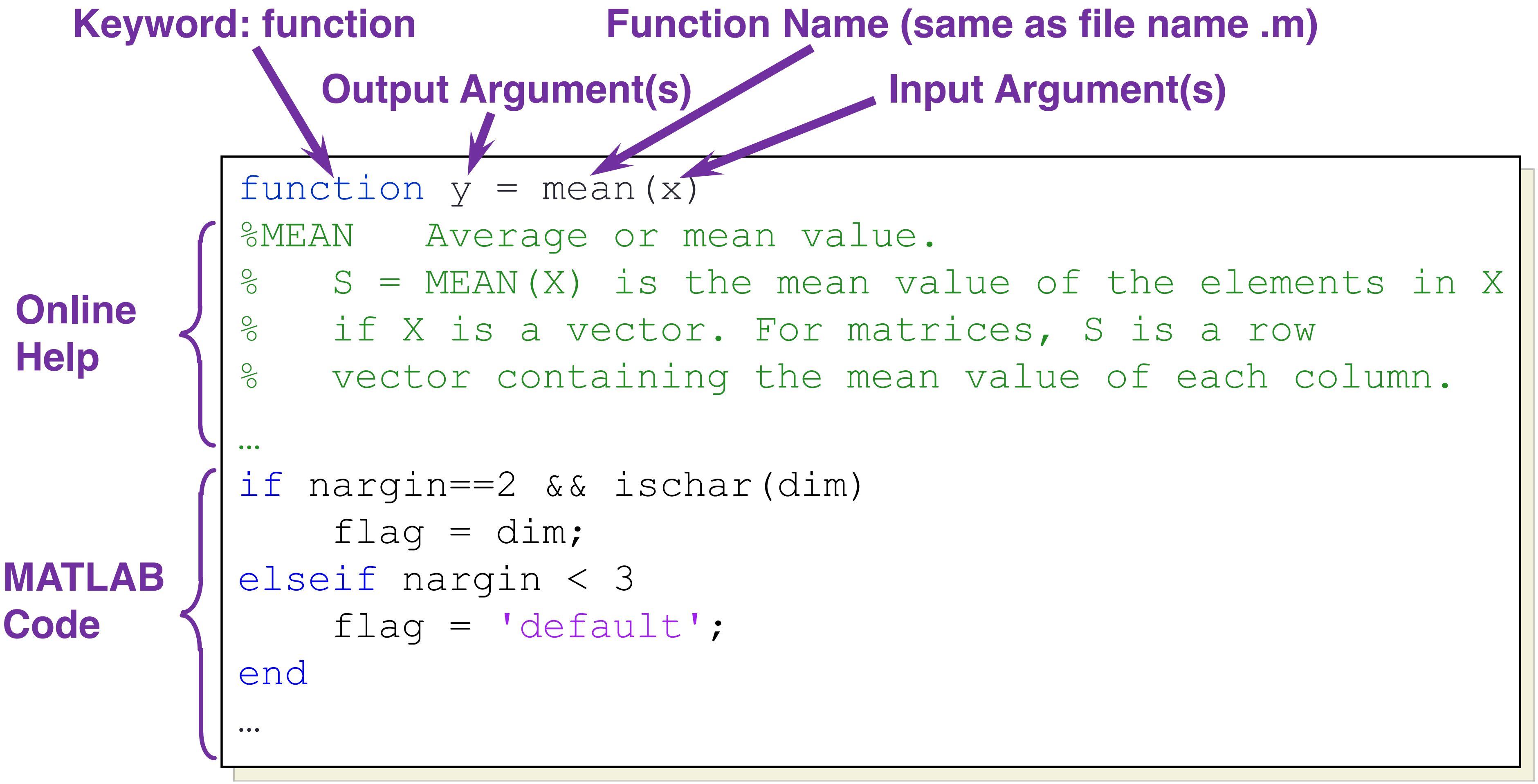
Scripts vs. Functions

- Scripts and functions are both `.m` files that contain MATLAB commands
- Functions are written when we need to perform routines

Scripts	Functions
• No input arguments • No output arguments • Operate on data in the global workspace	• Yes input arguments • Yes output arguments • Operate on data in the local workspace

Content of MATLAB Built-in Functions

```
>> edit(which('mean.m'))
```



Keyword: function

Output Argument(s)

Function Name (same as file name .m)

Input Argument(s)

Online Help

MATLAB Code

```
function y = mean(x)
%MEAN   Average or mean value.
%   S = MEAN(X) is the mean value of the elements in X
%   if X is a vector. For matrices, S is a row
%   vector containing the mean value of each column.

…
if nargin==2 && ischar(dim)
    flag = dim;
elseif nargin < 3
    flag = 'default';
end
…
```

Some Observations

- Keyword: `function`
- Function name matches the file name
- Directory: MATLAB needs to find the function
- Input and output variables are optional
- Local variables: `dim` and `flag` cannot be accessed

User Define Functions

- Write a function that calculate the displacement of free falling for given initial displacement x_0, initial velocity v_0, and duration of falling t:

$$x = x_0 + v_0 t + \frac{1}{2} g t^2$$

```
function x = freebody(x0,v0,t)
% calculation of free falling
% x0: initial displacement in m
% v0: initial velocity in m/sec
% t: the elapsed time in sec
% x: the depth of falling in m
x = x0 + v0.*t + 1/2*9.8*t.*t;
```

Functions with Multiple Inputs and Outputs

- The acceleration of a particle and the force acting on it are as follows:

$$a = \frac{v_2 - v_1}{t_2 - t_1}$$

$$F = ma$$

```
function [a F] = acc(v2,v1,t2,t1,m)
a = (v2-v1)./(t2-t1);
F = m.*a;
```

```
[Acc Force] = acc(20,10,5,4,1)
```

Exercise

- Write a function that asks for a temperature in degrees Fahrenheit
- Compute the equivalent temperature in degrees Celsius
- Show the converted temperature in degrees Celsius
- The script should keep running until no number is provided to convert
- You may want to use these functions:

```
input, isempty, break, disp, num2str
```

Function Default Variables

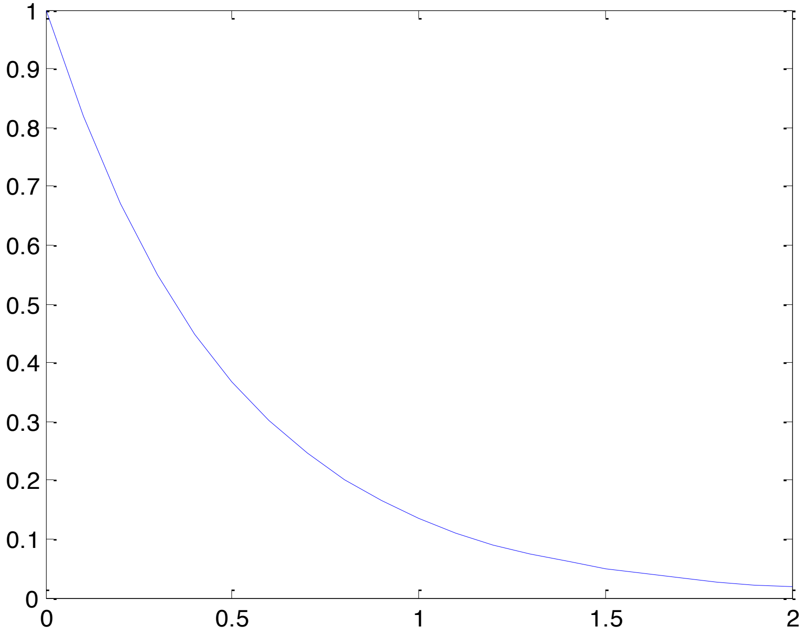
inputname	Variable name of function input
mfilename	File name of currently running function
nargin	Number of function input arguments
nargout	Number of function output arguments
varargin	Variable length input argument list
varargout	Variable length output argument list

```
function [volume]=pillar(Do,Di,height)
if nargin==2,
    height=1;
end
volume=abs(Do.^2-Di.^2).*height*pi/4;
```

Function Handles

- A way to create anonymous functions, i.e., one line expression functions that do not have to be defined in `.m` files

```
f = @(x) exp(-2*x);
x = 0:0.1:2;
plot(x, f(x));
```



End of Class