Introduction to Computer Programming for Engineers

EE050IU

Lecture 1 – Introduction

Introduction to Computer for Engineers

- Course Instructor: Dr. Nguyen Ngoc Truong Minh
- Position: Head of Dept. of Telecommunications
- Class room: LA1.301
- Class periods: Tuesday morning, 1-3
- Class hours: 08:00AM – 10:30AM

Introduction to Computer for Engineers

- Grading Policy
  - Attendance + Quiz (15%)
  - Homework Problems (15%)
  - Mid-term exam (25%)
  - Final Exam + Final Project (45%)
- Office hours: Monday 08:00AM-11:00AM, @A2.206
- Contact information: nntminh@hcmiu.edu.vn
- Lecture Notes and HW:
  - BLACKBOARD
  - Class hand-outs

Introduction to Computer for Engineers

- Classroom Policy
  - No Talking/Disturbing
  - No Cellphone during class
  - Quizzes will be given without notice
  - Attendance is required (80%)
  - Students missing 03 classes will not be allowed to take the final exam

What do we want you to learn ?

On the completion of this course, we expect you…

- To understand the fundamental programming concepts and solving problems in engineering
- To become proficient in the programming environment MATLAB/Octave
- To have fun ☺

Course Reference Materials

- MATLAB®
  - MATLAB Programming for Engineers (Stephen J. Chapman), Thompson Books
  - Introduction to MATLAB 7 for Engineers (William J. Palm III), McGraw Hill Books
  - MATLAB online help
  - Course notes on web page

Course Software

- MATLAB®
  - Student Version ($55)
    - Same as professional version without Toolboxes
    - Includes Symbolic Toolbox (Maple)
  - Available on Campus Computers
    - Professional version ($2,350-Perpetual, $940-Annual)
    - Many Toolboxes

Course Software Replacement

- Octave
  - Powerful mathematics-oriented syntax with built-in 2D/3D plotting and visualization tools
  - Free software, runs on GNU/Linux, macOS, BSD, and Microsoft Windows
  - Drop-in compatible with many MATLAB scripts

Topics Covered

- Introduction (Lecture 1)
- MATLAB/Octave (Lecture 2)
- Matrices and Vectors (Lecture 3)
- Mathematical Operations with Arrays (Lecture 4)
- Plots and Graphs (Lecture 5)
- Script and Function Files (Lecture 6)
- Logical Operators and Conditional Statements (Lecture 7)
- Project 1
- Iteration (Lecture 8)
- Loops and Strings (Lecture 9)
- Strings and Files (Lecture 10)
- Cell Arrays (Lecture 11)
- Curve Fitting and Interpolation (Lecture 12)
- Numerical Integration (Lecture 13)
- Graphical User Interface (GUI) (Lecture 14)

} Total of 14 classes

Graded Items

- Homework
  - – Due every week
  - – Typically, Octave programming
- In-class quizzes will be based on HW problems
- Projects
  - – More complicated programming assignments
  - – The final project will be a more extensive programming task
- Exams two exams: one midterm and one final

Course Grading Breakdown

Grading Breakdown

Attendances + Quizzes	15%
HWs	15%
Midterm Exam	25%
Final Exam+ Final Project	45%

Total	100%

Collaboration

- Homework/Projects are geared to help you learn and master the material. Do not mind if you work together. Indeed, encouraging an open and collaborative learning environment for homework and projects.
- Learning from each other is a GOOD thing! Copying from each other is a BAD thing! You must write your own code. That is the ONLY way to learn.
- In class, Projects, Quizzes as well as the Midterm and Final Exam are used to give the majority of the assessment of your ability. If you do not do your own work on the HW/Projects, you will not pass the exams and quizzes.

Some Tips to Learn

- Learn by doing. Always play with the code while learning
- Seek out more online resources. There's a wealth of content over there!
- Don't just read the sample code. Tinker with it!
- Take breaks when debugging
  - "The most important thing in computer programming is SYNTAX"
  - "Programming is NOT a spectator sport. To become good, you must practice - practice - and practice"

Don’t wait until the last minute to get help

You CANNOT slack off in this class, even for a few days

Introduction to MATLAB/Octave

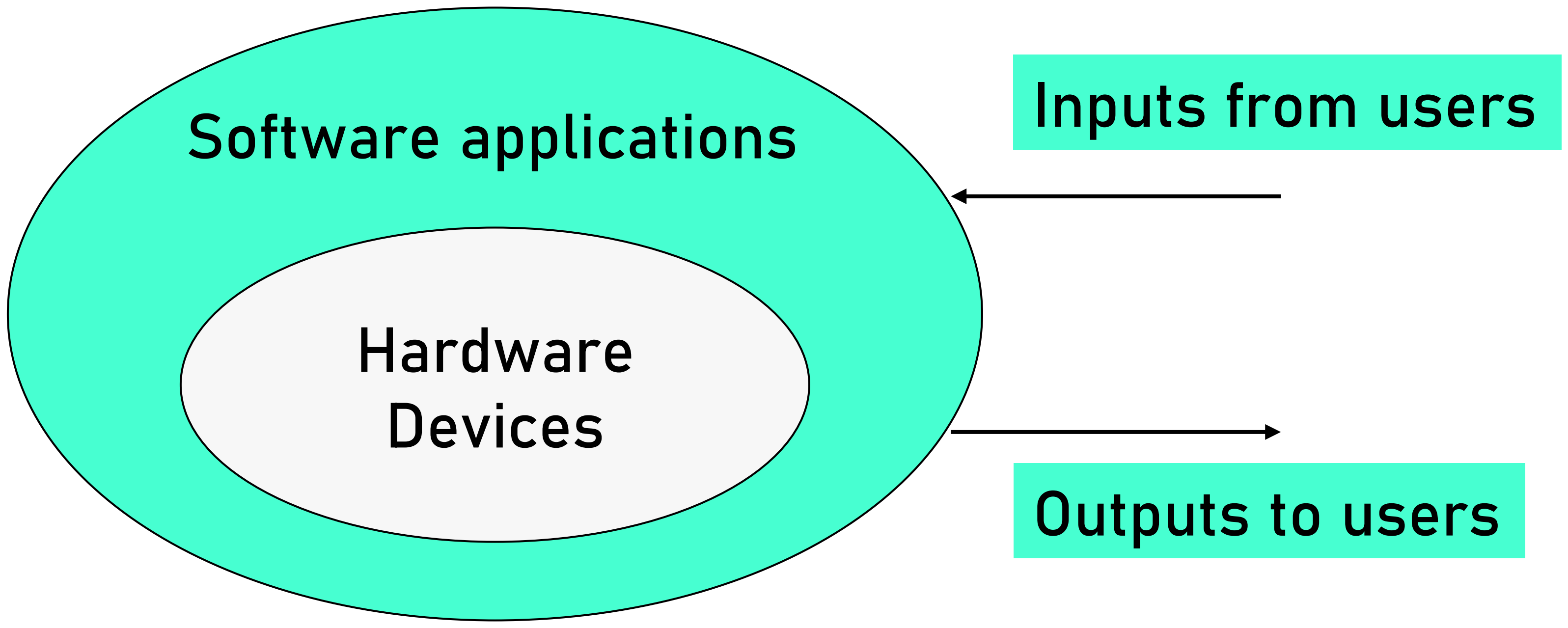
GNU Octave

https://www.octave.org/

How and why do engineers need computers to solve problems?

- We all know that engineers (as well as everyone else) use computers to communicate (email), write reports (word, excel), make presentations (power point), conduct research (web), do their taxes, check the football scores, create virtual world, play games…
- That is not what we talking about here! Given a technical problem to solve how and why do engineers use computers to solve those problems?

Computer Systems



Software: What is Programming?

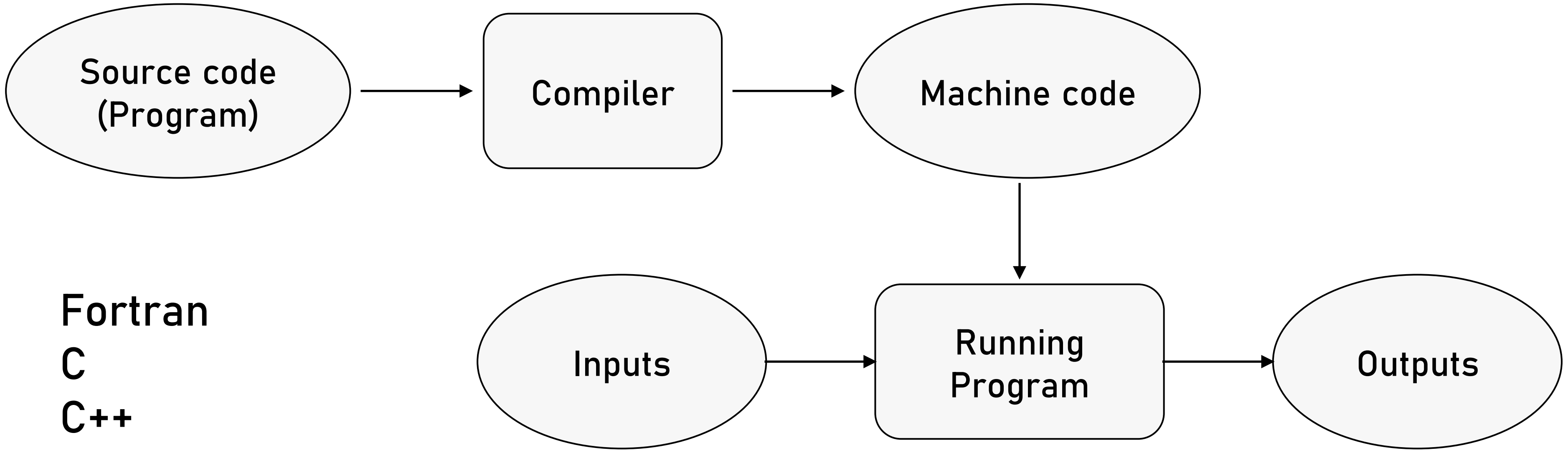
Programming is the process of

- Analyzing a problem
- Designing an algorithmic solution (i.e. coming up with a recipe or list of rules)
- Coding the design in a particular programming language
- Testing the correctness of the solution
- Maintaining the solution (receiving user feedback and modifying the design and code)

Programming Languages

- A programming language is a way for you to take your computer algorithm and put it in a form that the computer can understand.
- Like any other language, a computer language has a vocabulary you must learn (commands).
- Like any other language, a computer language has a list of rules on how those commands can be combined together.
- Like any other language, learning a computer language takes time and effort.
- Unlike any other language, however, computer languages are not forgiving on how you apply the rules or spell the commands (syntax). If you do not get the syntax right, the programming will not run properly.

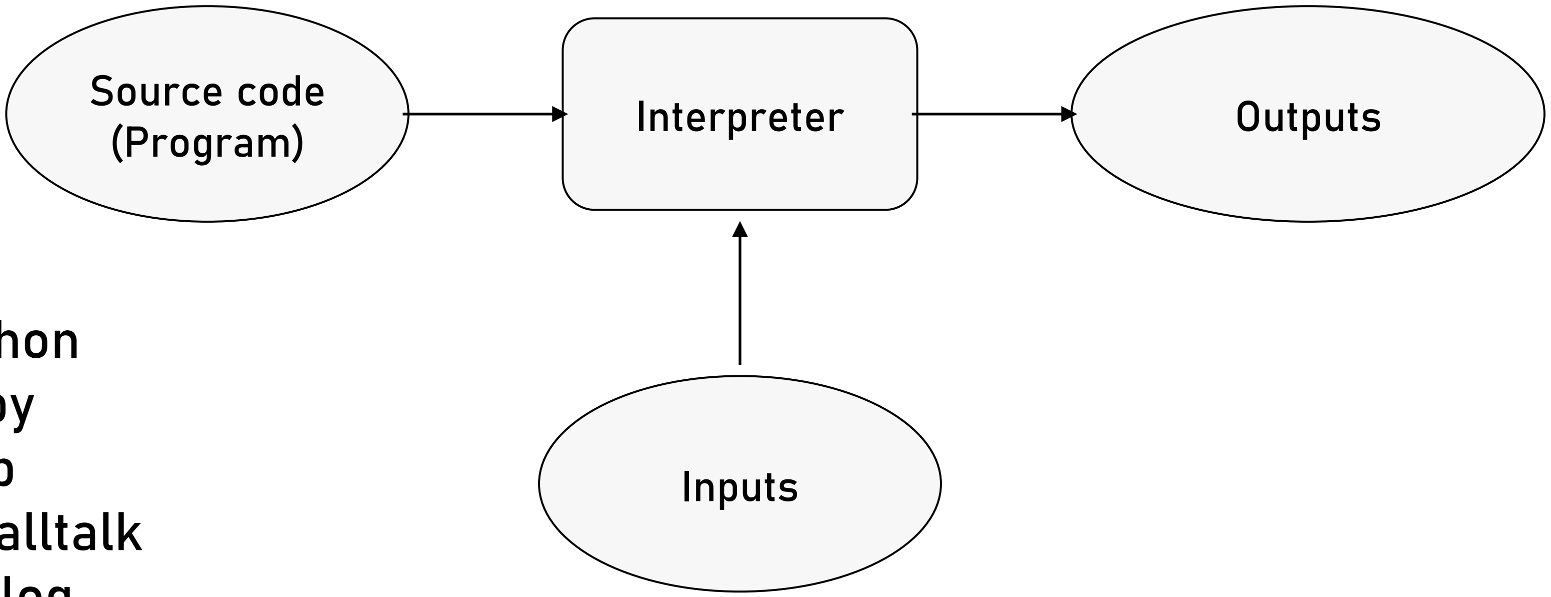
Translate and Run a Program



Fortran
C
C++
C#
Java

Compiled languages require a separate compilation step before program execution

Translate and Run a Program



Python
Ruby
Lisp
Smalltalk
Prolog
Matlab

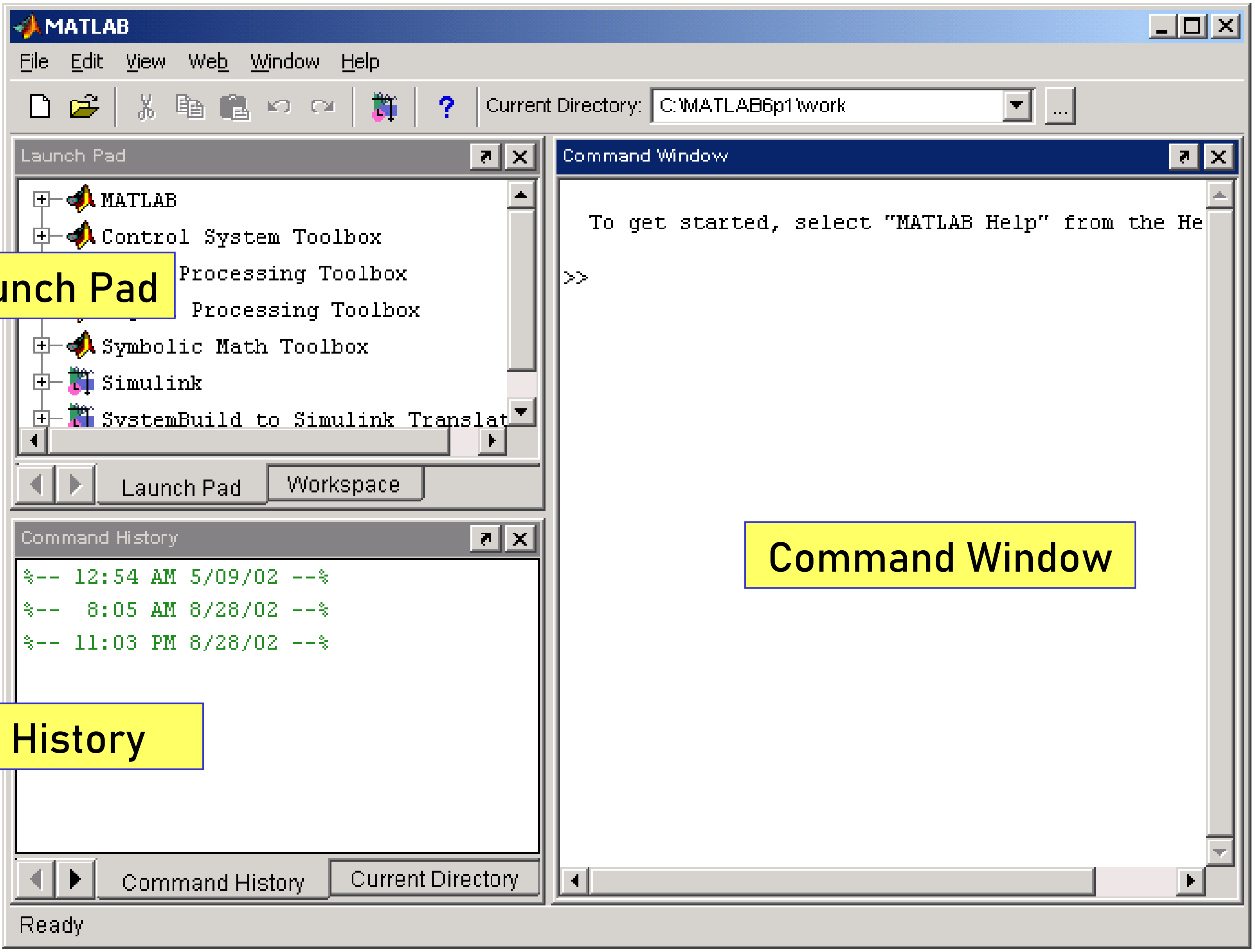
Interpreted languages translate a program during execution

What is MATLAB?

- MATLAB was originally developed to be a "matrix laboratory," written to provide easy access to matrix software developed by the LINPACK and EISPACK projects.
- Since then, the software has evolved into an interactive system and programming language for general scientific and technical computation and visualization.

Ladies and Gentlemen, please start your computers

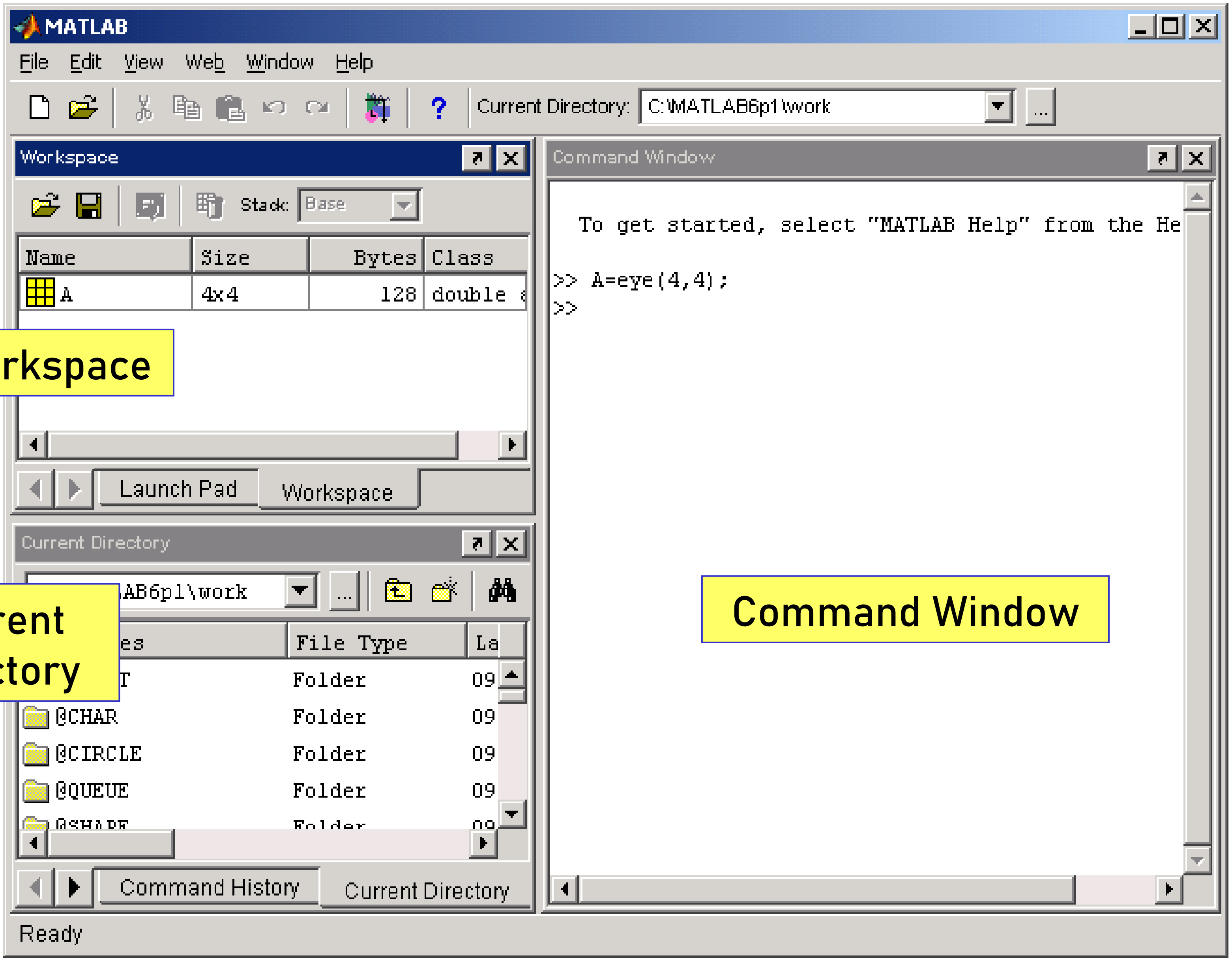
MATLAB/Octave Desktop



```
%-- 12:54 AM 5/09/02 --%
%--  8:05 AM 8/28/02 --%
%-- 11:03 PM 8/28/02 --%
```

```
  To get started, select "MATLAB Help" from the He

>>
```

MATLAB/Octave Desktop – cont'd

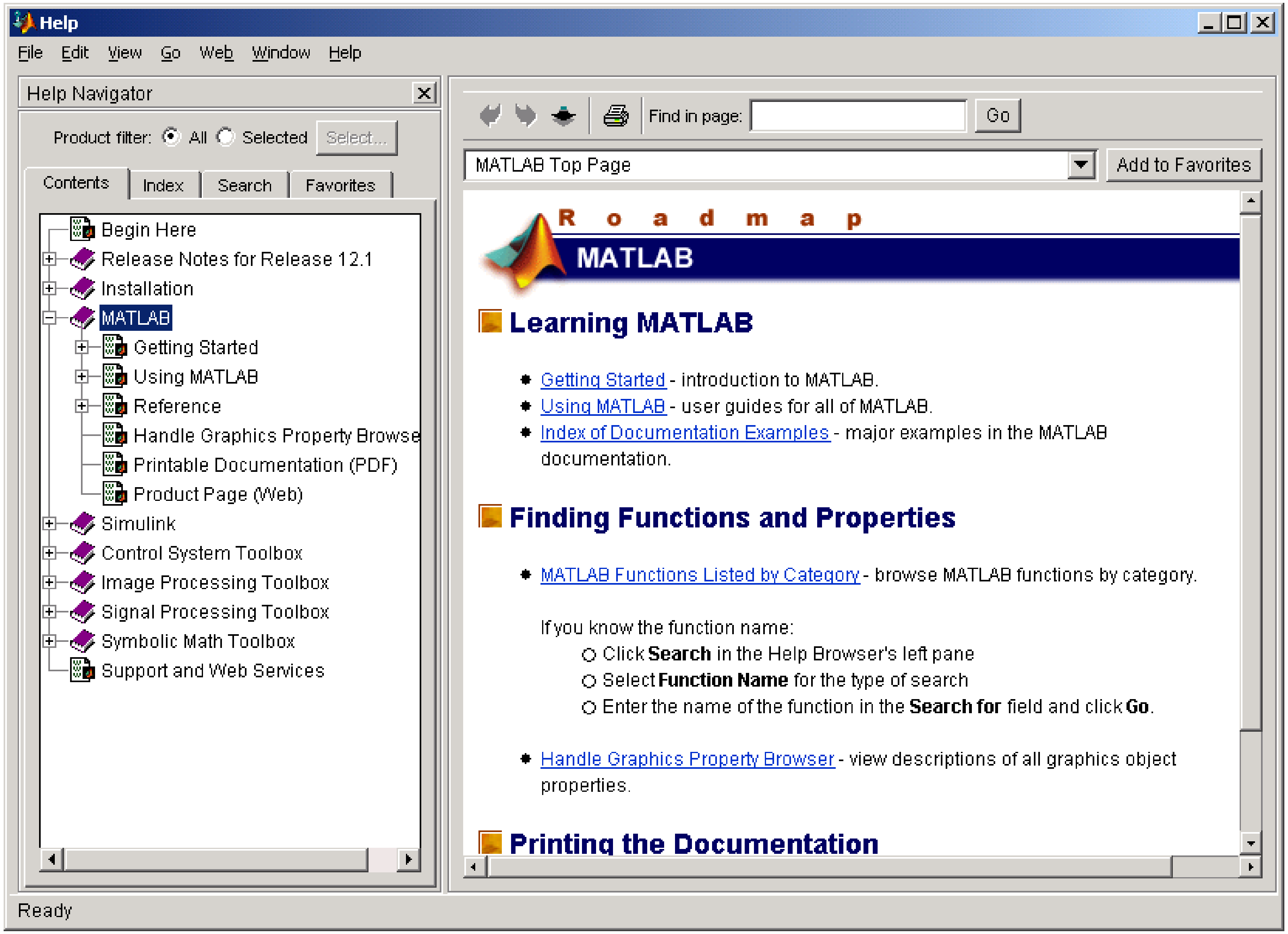


Name	Size	Bytes	Class
A	4x4	128	double

File Type	La
Folder	09
Folder	09
Folder	09
Folder	09

```
 To get started, select "MATLAB Help" from the He

>> A=eye(4,4);
>>
```

MATLAB/Octave Help


Help
File Edit View Go Web Window Help
Help Navigator
Product filter: [✓ All] Selected Select...
Contents Index Search Favorites
Find in page: Go
MATLAB Top Page
Add to Favorites
Ready

Roadmap

MATLAB

Learning MATLAB

- Getting Started - introduction to MATLAB.
- Using MATLAB - user guides for all of MATLAB.
- Index of Documentation Examples - major examples in the MATLAB documentation.

Finding Functions and Properties

- MATLAB Functions Listed by Category - browse MATLAB functions by category.

  If you know the function name:
  - Click **Search** in the Help Browser's left pane
  - Select **Function Name** for the type of search
  - Enter the name of the function in the **Search for** field and click **Go**.

- Handle Graphics Property Browser - view descriptions of all graphics object properties.

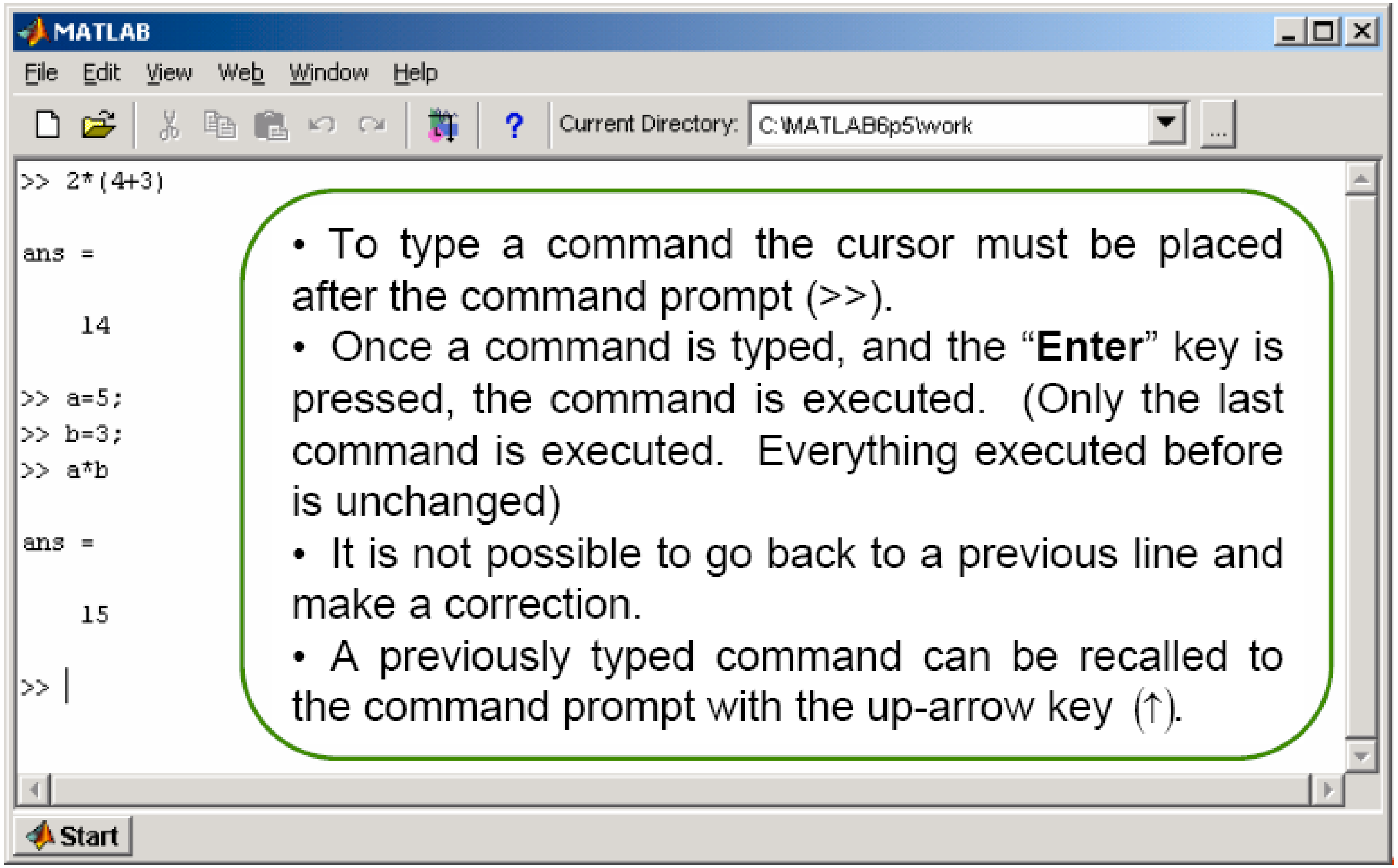
Printing the Documentation

The MATLAB/Octave Environment

- MATLAB is an interpreted language
  - Commands are typed into the COMMAND Window and executed immediately
  - Variables are allocated in memory as soon as they are first used in an expression
  - Commands must be re-entered to be re-executed
- All variables created in the Command Window are in what is called the Base Workspace
  - Variables can be reassigned new values as needed
  - Variables can be selectively cleared from the workspace
- The Workspace can be saved to a data file
  - File extension is .mat (ex: mydata.mat)
  - File is in binary and essentially unreadable by humans
  - .mat files can be reloaded back into the MATLAB Workspace

FIRST TRY: USING MATLAB/Octave LIKE A CALCULATOR

- Before we begin writing MATLAB/Octave programs, lets first get use to it by using it as a very powerful (and expensive) calculator.


MATLAB
File Edit View Web Window Help
Current Directory: C:\MATLAB6p5\work
Start

```
>> 2*(4+3)

ans =

    14

>> a=5;
>> b=3;
>> a*b

ans =

    15

>> |
```

- To type a command the cursor must be placed after the command prompt (>>).
- Once a command is typed, and the "**Enter**" key is pressed, the command is executed. (Only the last command is executed. Everything executed before is unchanged)
- It is not possible to go back to a previous line and make a correction.
- A previously typed command can be recalled to the command prompt with the up-arrow key (↑).

Basic Math

ARITHMETIC OPERATIONS WITH SCALARS

Operation	Symbol	Example
Addition	+	5+3
Subtraction	-	5-3
Multiplication	*	5*3
Right Division	/	5/3
Left Division	\	5\3=3/5
Exponentiation	^	5^3

NOTE: For scalars the arithmetic operations are the usual ones. For vectors and matrices the arithmetic operations can either follow matrix algebra rules, or can be performed on element-by-element basis (discussed in the next lectures).

Order of Precedent

(The order in which operations are executed by the computer)

Higher-precedence operations are executed before lower-precedence operations.

If two operations have the same precedence, then the expression is executed from left to right.

PRECEDENCE	**OPERATION**
First	Parentheses, starting with the innermost pair.
Second	Exponentiation.
Third	Multiplication and division (equal precedence).
Fourth	Addition and subtraction (equal precedence).

FIRST TRY: USING MATLAB/Octave LIKE A CALCULATOR

Using numbers:

```
>> 7+8/2
ans =
   11
```

Type and press **Enter**

Computer response

8/2 is executed first

```
>> (7+8)/2
ans =
   7.5000
```

Type and press **Enter**

Computer response

7+8 is executed first

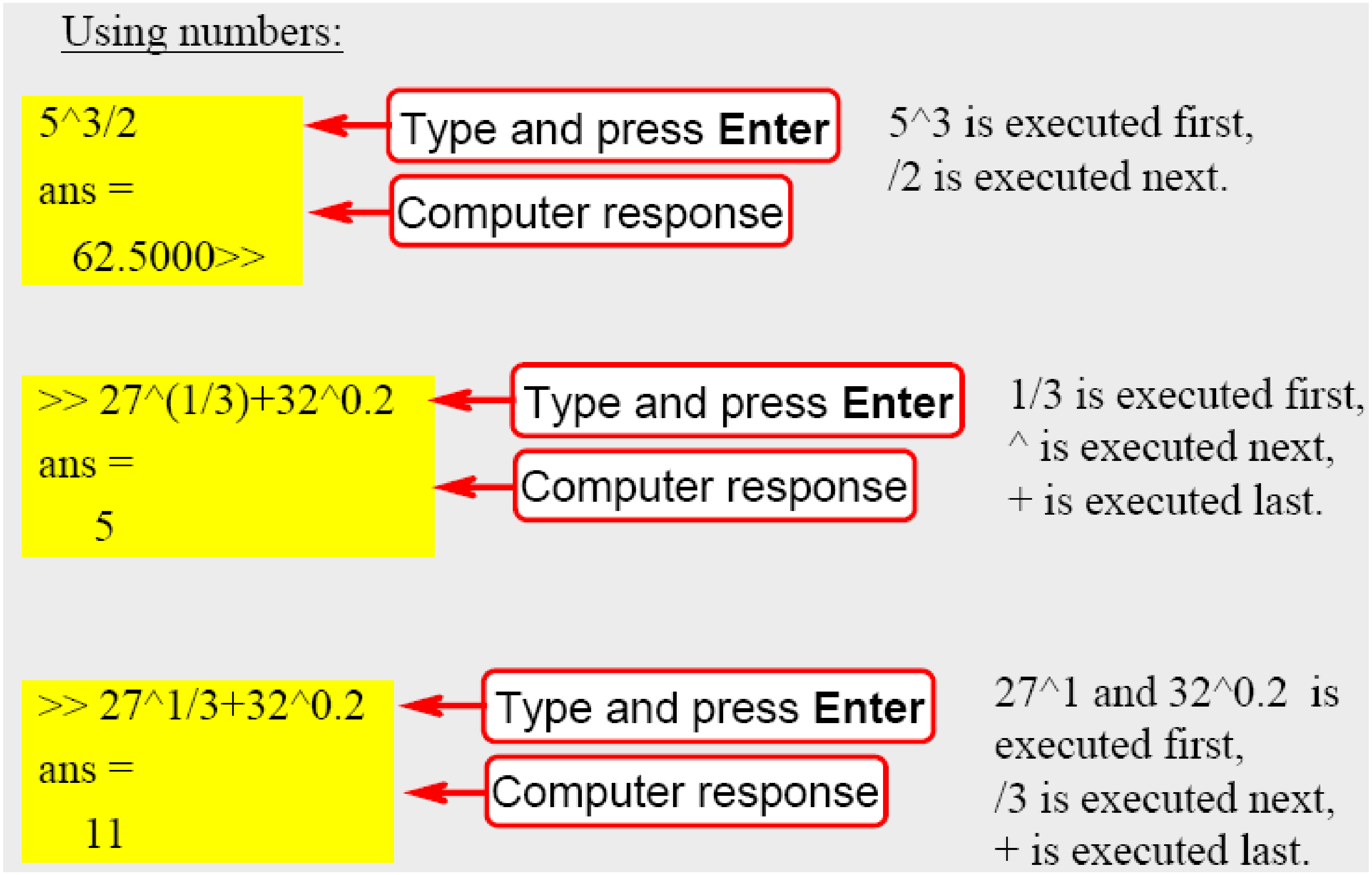
```
>> 4+5/3+2
ans =
   7.6667
```

Type and press **Enter**

Computer response

5/3 is executed first

FIRST TRY: USING MATLAB/Octave LIKE A CALCULATOR



Using numbers:

```
5^3/2
ans =
   62.5000>>
```

Type and press **Enter**

Computer response

5^3 is executed first,
/2 is executed next.

```
>> 27^(1/3)+32^0.2
ans =
    5
```

Type and press **Enter**

Computer response

1/3 is executed first,
^ is executed next,
+ is executed last.

```
>> 27^1/3+32^0.2
ans =
   11
```

Type and press **Enter**

Computer response

27^1 and 32^0.2 is executed first,
/3 is executed next,
+ is executed last.

Data Type

• int8	8-bit signed integer (-127 to 128)
• uint8	8-bit unsigned integer (0 to 255)
• int16	16-bit signed integer (-32767 to 32768)
• uint16	16-bit unsigned integer (0 to 65536)
• int32	32-bit signed integer (- 2.1475e+09 to 2.1475e+09)
• uint32	32-bit unsigned integer (0 to 4.2950e+09)
• int64	64-bit signed integer
• uint64	64-bit unsigned integer
• single	single precision numerical data (32 bit)
• double	double precision numerical data (64 bit)
• logical	logical values of 1 or 0, represent true and false

Display Format

The `format` command controls how output numbers appear on the screen. Input numbers can be written in any format.

`format short`	(the default)
41.4286	Fixed-point with 4 decimal digits.
`format long`	
41.42857142857143	Fixed-point with 14 decimal digits.
`format short e`	
4.1429e+001	Scientific with 4 decimal digits.
`format long e`	
4.142857142857143e+001	Scientific with 15 decimal digits.
`format bank`	
41.43	Two decimal digits.

format rat
4/3

Rational fraction

MATLAB/Octave built-in functions

In addition to arithmetic operations, MATLAB can be used to calculate elementary math functions. The most common ones are:

`sin(x)`	`x` in radians	`exp(x)`	exponential
`cos(x)`	`x` in radians	`log(x)`	natural logarithm
`tan(x)`	`x` in radians	`log10(x)`	base 10 logarithm
`cot(x)`	`x` in radians	`sqrt(x)`	square root
		`abs(x)`	absolute value

The inverse is: `asin(x)`, `acos(x)`, etc.

Examples:

```
>> sin(0.78539)

ans =

   0.7071
```

```
>> sqrt(169)

ans =

   13
```

```
>> log10(10000)

ans =

   4
```

MATLAB has hundreds of built-in functions (this will be discussed in future lectures).

Assignment Operator

In MATLAB, the = sign is called the ASSIGNMENT OPERATOR.

The ASSIGNMENT OPERATOR assigns a value to a variable.

Variable = A value, or a computable value

The left hand side can only be **<u>one</u>** variable.

The right hand side can be a specific value, or a computable expression (an expression that includes values and/or previously defined variables).

Assignment Operator

For example, if you type:

```
>> x = 3

x =

     3
```

MATLAB assigns the value of 3 to x.

If you then type:

```
>> x = x + 5

x =

     8
```

MATLAB assigns a new value to x, which is the old value 3 plus 5.

(In mathematics this expression has no meaning since it implies: 0 = 5.)

However, the statement:

x + 4 = 30 <u>is not valid</u>. MATLAB does not solve for x,

but the statement:

x = 30 – 4 is valid (the number 26 is assigned to x.)

Defining Variables

A variable is defined by typing a variable name followed by the assignment operator (equal sign) and then a value, or a mathematical expression.

```
>> a=8
a =
    8
```

Type and press **Enter**

Computer response

```
>> B=12
B =
   12
```

Type and press **Enter**

Computer response

Once a variable is defined, the computer remembers and stores its value. The variable can then be used in further calculations.

```
>> a+B
ans =
    20
```

```
>> a/B
ans =

0.6667
```

```
>> B/a
ans =
     1.5000
```

```
>> B^a
ans =

429981696
```

Defining Variables

Variables can also be used to define new variables

```
>> d=a*B
d =
    96
```

Once in existence, variables can be used in functions

```
>> sqrt(d)
ans =
    9.7980
```

A previously defined variable can be redefined and reassigned a new value.

Rules About Variables

- Variable names can be up to 63 characters long.
- Variable names **must** begin with a letter.
- Variable names can contain letters, digits, and the underscore character.
- MATLAB is case sensitive; it distinguishes between **UPPERCASE** and **lowercase** letters.

  For example, `A` and `a` are not the same variable.

Rules About Variables

AVOID USING NAMES OF FUNCTIONS FOR VARIABLES.

Once a function name is used to define a variable, the function cannot be used without quitting Matlab and restarting it.

This means that variables should not be named

sin, cos, exp, tan, sqrt, ……., etc.

OR:

max, min, sum, det, …., etc.

Predefined Variables

MATLAB has several variables that are predefined.
These variables can be redefined to have any other value.
It is probably better not to use the predefined variables as variable names.

Some of the predefined variables are:

`pi`	(π)
`eps`	(the smallest difference between two numbers)
`inf`	(infinity)
`i`	(square root of -1),
`j`	(square root of -1)
`ans`	(the value of the most recent calculation)

Typing these variables gives:

```
>> pi
ans =
    3.1416
```

```
>> sin(pi/4)
ans =
    0.7071
```

```
>> eps
ans =
    2.2204e-016
```

```
>> inf
ans =
    Inf
```

```
>> i
ans =
        0 + 1.0000i
```

Some Useful Commands

When these commands are typed in the Command Window they either provide information, or perform a task.

`clc`	Clears the command window.
`clear`	Clears all variables from memory.
`clear x y z`	Clears only variables x, y and z.
`clf`	Clears the Figure Window.
`who`	Lists the variables currently in memory.

MATLAB/Octave Variables

- MATLAB/Octave is a fully-functional programming language
- This means we get variables
  - name = value
    - Name can be anything made of letters, numbers, and a few symbols (_). Must start with a letter
  - End line with “;” to avoid output
    - Can also use “;” to put multiple commands on a line
  - List with who
  - Delete with clear
  - More info with whos

Reserved Words…

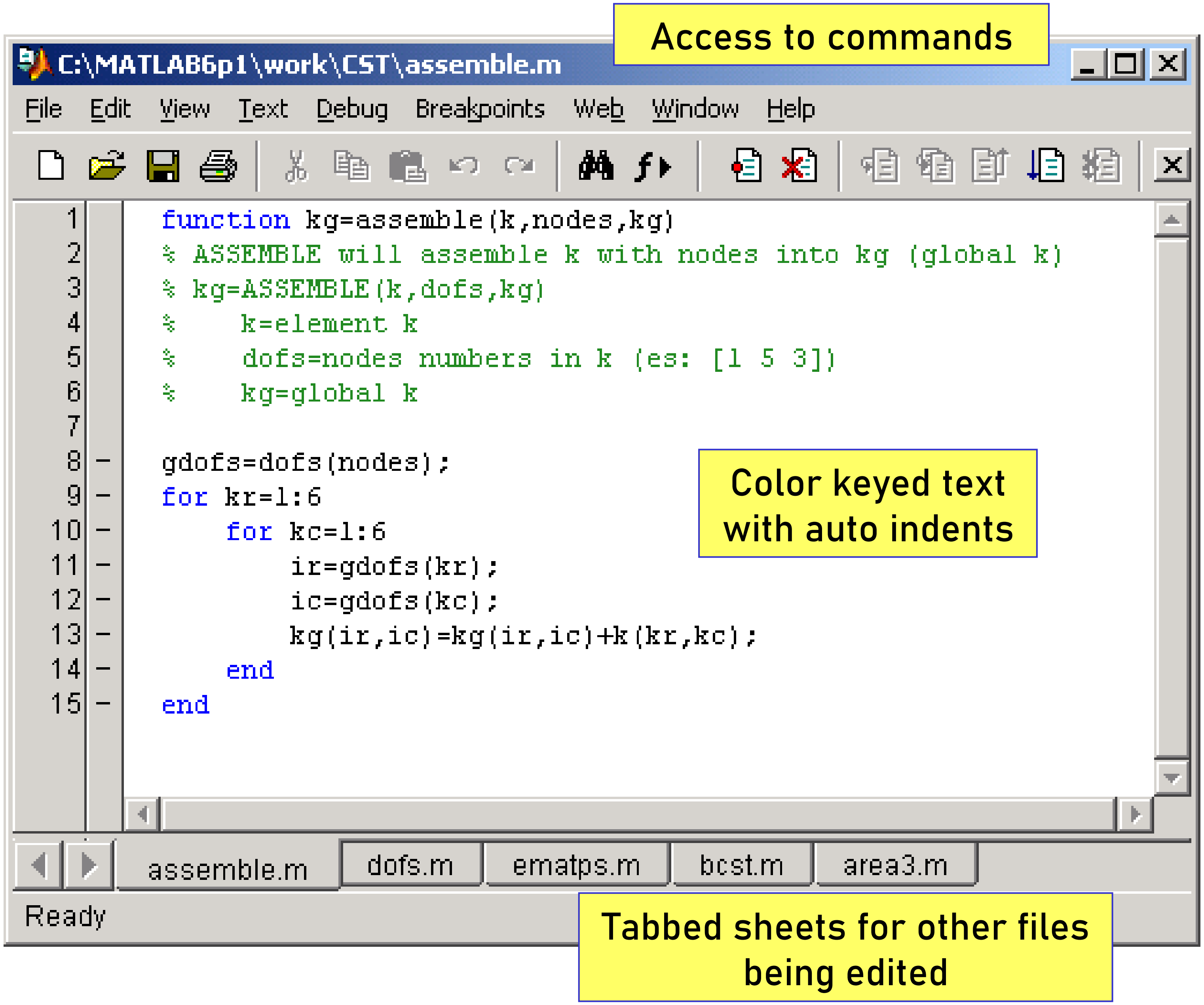
- MATLAB/Octave has some special (reserved) words that you may not use…

for	switch
end	continue
if	else
while	try
function	catch
return	global
elif	persistent
case	break
otherwise	

MATLAB/Octave Programs

- MATLAB/Octave is an extravagant calculator if all we can do is execute commands typed into the Command Window…
- So how can we execute a "program?"
- Programs in MATLAB/Octave are:
  - Scripts, or
  - Functions
- Scripts: MATLAB/Octave statements that are fed from a file into the Command Window and executed immediately
- Functions: Program modules that are passed data (arguments) and return a result (i.e., sin(x))
- These can be created in any text editor (but MATLAB/Octave supplies a nice built-in editor)

MATLAB/Octave Editor


C:\MATLAB6p1\work\CST\assemble.m
File Edit View Text Debug Breakpoints Web Window Help
assemble.m
dofs.m
ematps.m
bcst.m
area3.m
Ready

```
function kg=assemble(k,nodes,kg)
% ASSEMBLE will assemble k with nodes into kg (global k)
% kg=ASSEMBLE(k,dofs,kg)
%     k=element k
%     dofs=nodes numbers in k (es: [1 5 3])
%     kg=global k

gdofs=dofs(nodes);
for kr=1:6
    for kc=1:6
        ir=gdofs(kr);
        ic=gdofs(kc);
        kg(ir,ic)=kg(ir,ic)+k(kr,kc);
    end
end
```

Access to commands

Color keyed text with auto indents

Tabbed sheets for other files being edited

The ';' and '%' operators

;

Typing a semi-colon, " ; " at the **end** of a line of input prevents the output from being displayed in the command window

For example

```
>> abc=37
abc =
    37
```

```
>> def=23;
>>
```

%

Typing a percent sign, "%", at the **beginning** of a line of input designates the line as a comment

Comments are not executed by Matlab but are included for documentation purposes

Examples

- *Ex.1 Write your first MATLAB/Octave program*
  - a = 3;
  - b = 5;
  - c = a + b
- *Output:*

```
8
```

Examples

- *Ex. 2 The meaning of "a = b"*
  - a = 3;
  - b = a;
  - b
- *Output:*

3

Examples

- *Ex.3 The meaning of "a = b", continued*
  - a = 3;
  - a = a+1;
  - a
- *Output:*

```
4
```

Examples

- *Ex.4 Basic math operations*
  - a = 3;
  - b = 9;
  - c = 2*a + b^2 - a*b + b/a - 10
- *Output:*

53

Examples

- *Ex.5 Formatted output*
  - fprintf('Hello')
- *Output:*

```
Hello
```

Examples

- *Ex.6 Formatted output*

```
a = 3;
b = a*a;
c = a*a*a;
d = sqrt(a);
fprintf('%4u square equals %4u \r', a, b)
fprintf('%4u cube equals %4u \r', a, c)
fprintf('The square root of %2u is %6.4f \r', a, d)
```

- ***Output:***

```
3 square equals 9
3 cube equals 27
The square root of 3 is 1.7321
```

HW 1

1. You throw a ball straight up in the air with an initial speed of 25 m/s. [g = 9.8 m/s²]

a. What is the maximum height the ball rise from the release point?

b. How long does it take to reach the highest point?

c. At what time(s) will it be 25 m above the release point?

HW 1

1. You throw a ball straight up in the air with an initial speed of 25 m/s. [g = 9.8 m/s²]

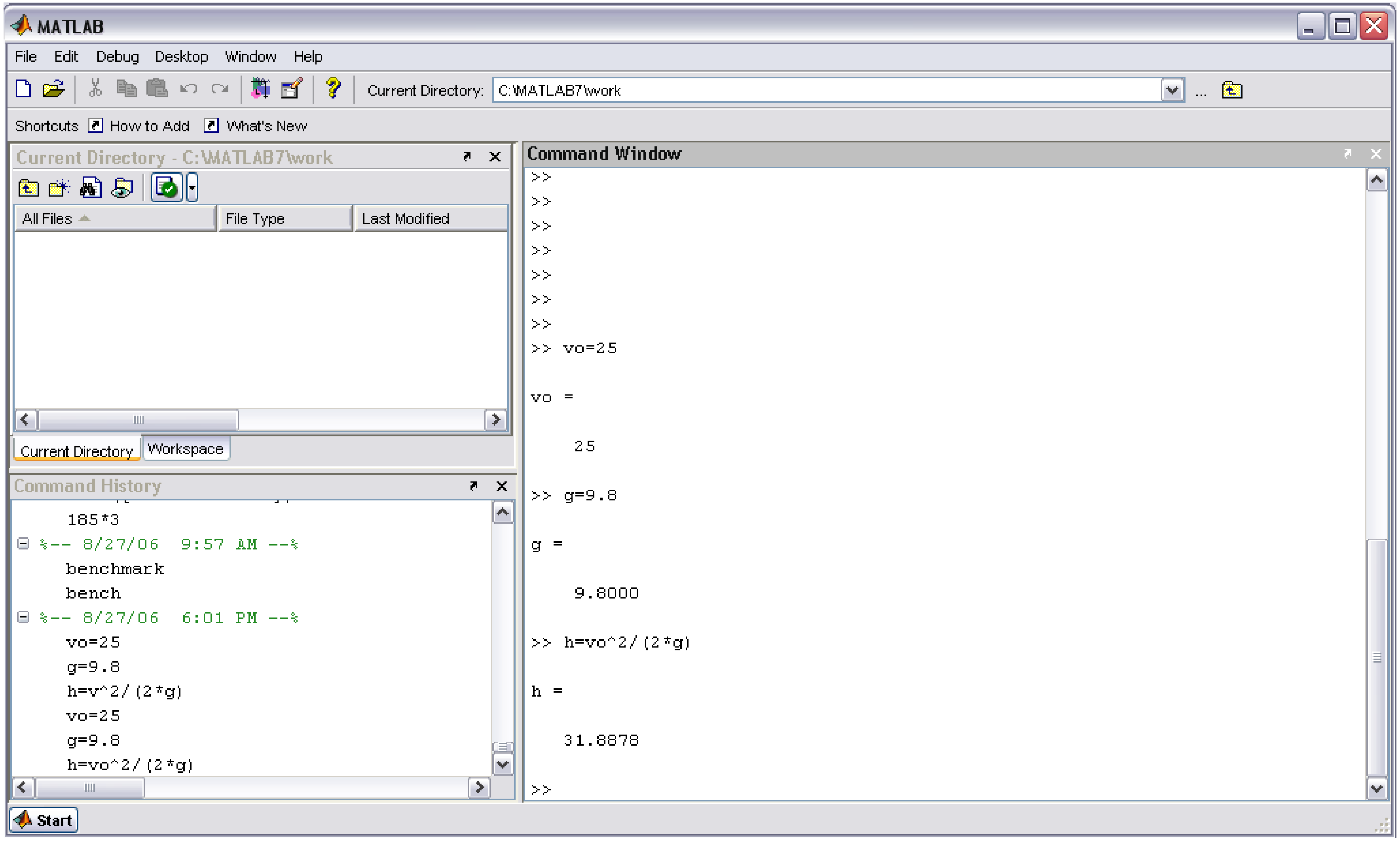
a. What is the maximum height the ball rise from the release point?

Solution:

$$\frac{1}{2}mv_o^{\,2} = mgh \quad \Rightarrow \quad h = \frac{v_o^{\,2}}{2g}$$

HW 1

(a) Solution from the command line



MATLAB
File Edit Debug Desktop Window Help
Current Directory: C:\MATLAB7\work
Shortcuts How to Add What's New
Current Directory - C:\MATLAB7\work
All Files File Type Last Modified
Current Directory Workspace
Command History
Command Window
Start

Command History:

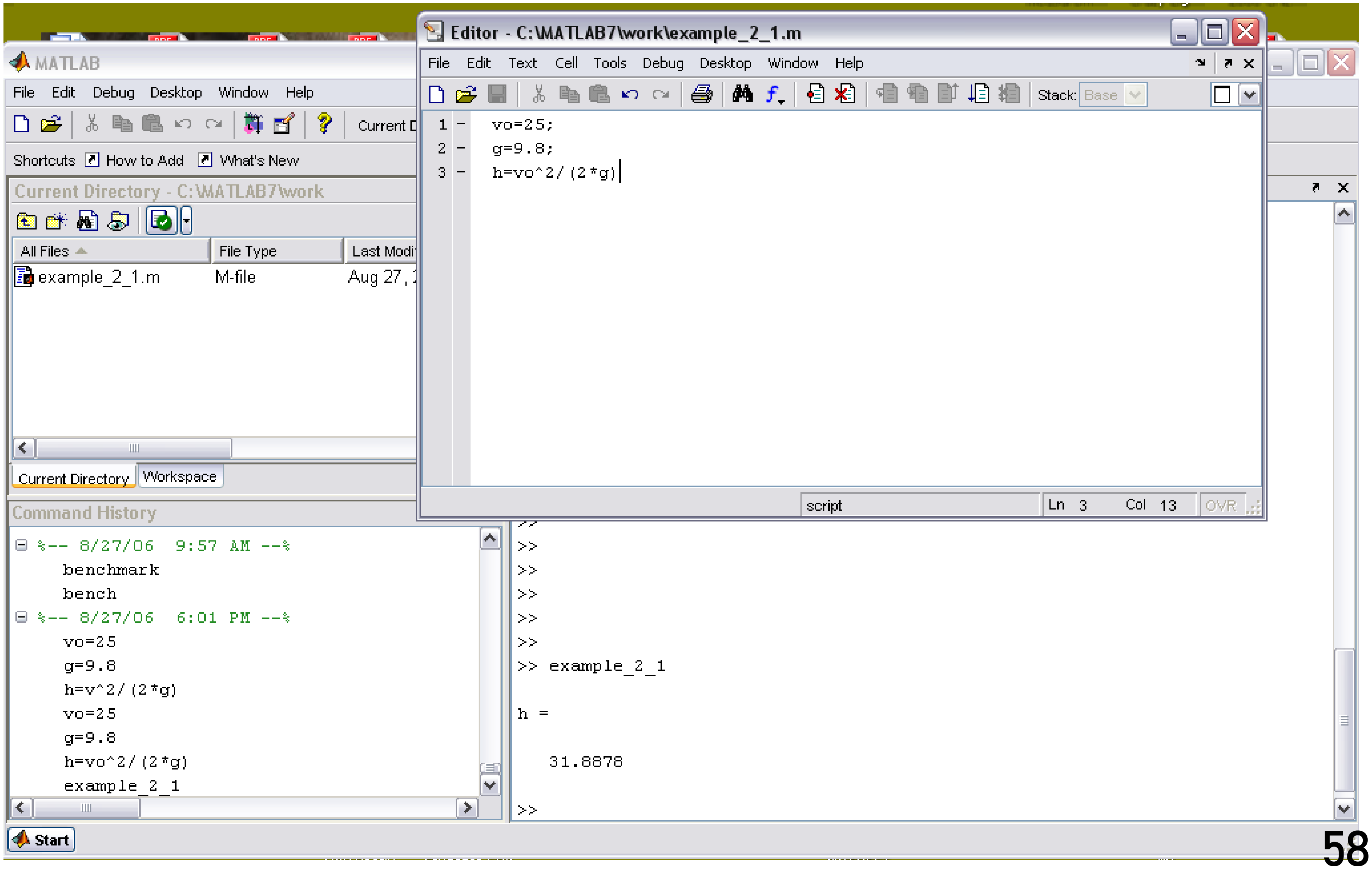
```
    185*3
%-- 8/27/06   9:57 AM --%
    benchmark
    bench
%-- 8/27/06   6:01 PM --%
    vo=25
    g=9.8
    h=v^2/(2*g)
    vo=25
    g=9.8
    h=vo^2/(2*g)
```

Command Window:

```
>>
>>
>>
>>
>>
>>
>>
>> vo=25

vo =

    25

>> g=9.8

g =

    9.8000

>> h=vo^2/(2*g)

h =

   31.8878

>>
```

HW 1

(b) Solution using a script file



Editor - C:\MATLAB7\work\example_2_1.m

```
vo=25;
g=9.8;
h=vo^2/(2*g)
```

Current Directory - C:\MATLAB7\work

All Files	File Type	Last Modi
example_2_1.m	M-file	Aug 27,

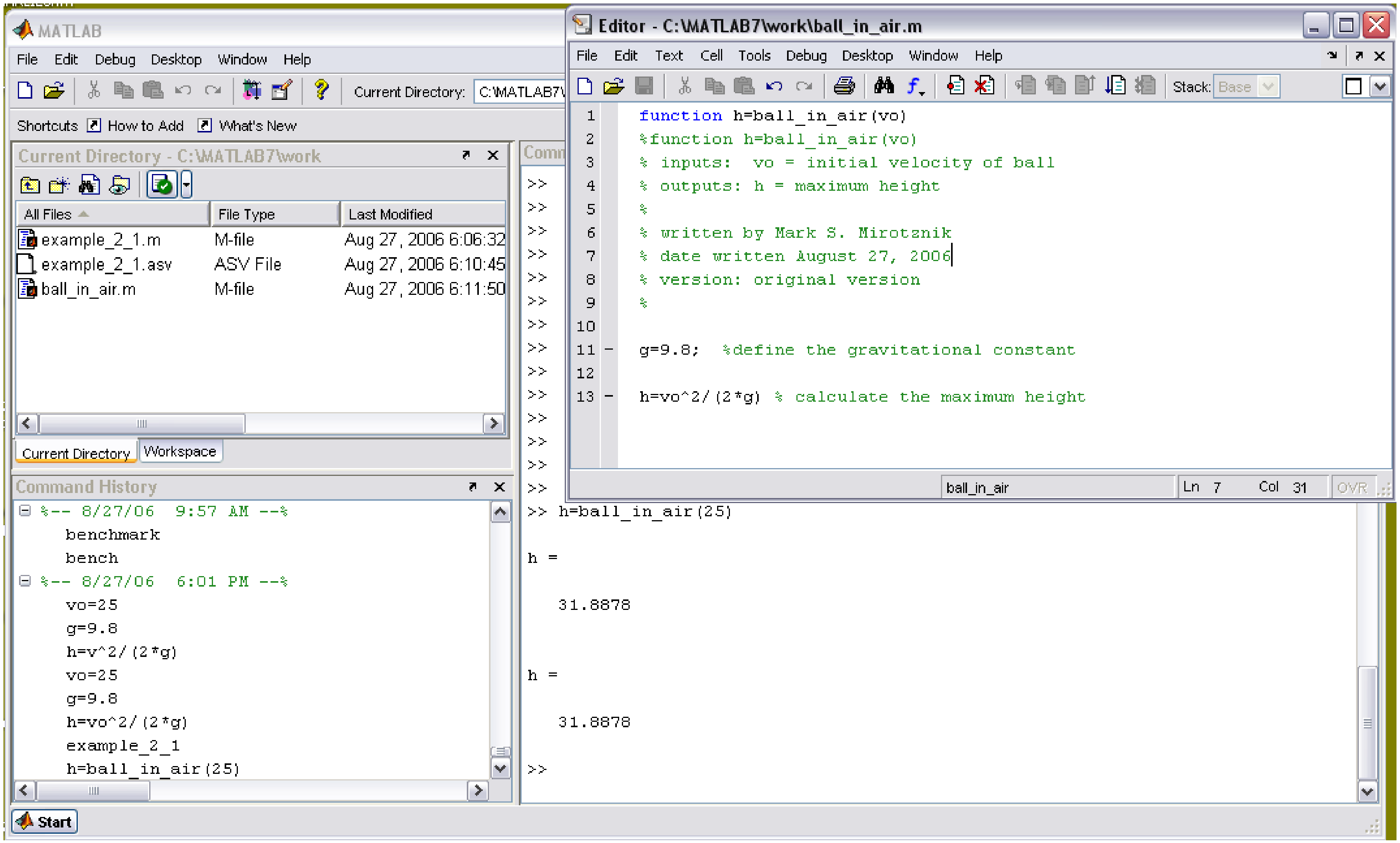
Command History

```
%-- 8/27/06   9:57 AM --%
    benchmark
    bench
%-- 8/27/06   6:01 PM --%
    vo=25
    g=9.8
    h=v^2/(2*g)
    vo=25
    g=9.8
    h=vo^2/(2*g)
    example_2_1
```

```
>>
>>
>>
>>
>>
>> example_2_1

h =

   31.8878

>>
```

HW 1

(c) Solution using a function



MATLAB
File Edit Debug Desktop Window Help
Current Directory: C:\MATLAB7\
Shortcuts How to Add What's New
Current Directory - C:\MATLAB7\work
Current Directory
Workspace
Command History
Start
Editor - C:\MATLAB7\work\ball_in_air.m
File Edit Text Cell Tools Debug Desktop Window Help
Stack: Base
ball_in_air
Ln 7 Col 31 OVR

All Files	File Type	Last Modified
example_2_1.m	M-file	Aug 27, 2006 6:06:32
example_2_1.asv	ASV File	Aug 27, 2006 6:10:45
ball_in_air.m	M-file	Aug 27, 2006 6:11:50

```
function h=ball_in_air(vo)
%function h=ball_in_air(vo)
% inputs:  vo = initial velocity of ball
% outputs: h = maximum height
%
% written by Mark S. Mirotznik
% date written August 27, 2006
% version: original version
%

g=9.8;  %define the gravitational constant

h=vo^2/(2*g) % calculate the maximum height
```

```
%-- 8/27/06  9:57 AM --%
   benchmark
   bench
%-- 8/27/06  6:01 PM --%
   vo=25
   g=9.8
   h=v^2/(2*g)
   vo=25
   g=9.8
   h=vo^2/(2*g)
   example_2_1
   h=ball_in_air(25)
```

```
>> h=ball_in_air(25)

h =

   31.8878


h =

   31.8878

>>
```

Hint

a. What is the maximum height the ball rise from the release point?

$$\frac{1}{2}mv_o^{\ 2} = mgh \quad \Rightarrow \quad h = \frac{v_o^{\ 2}}{2g}$$

b. How long does it take to reach the highest point?

$V_{final} = V_{initial} + gt$

c. At what time(s) will it be 25m above the release point?

$x = gt^2/2$

End of Lecture 1

Questions?