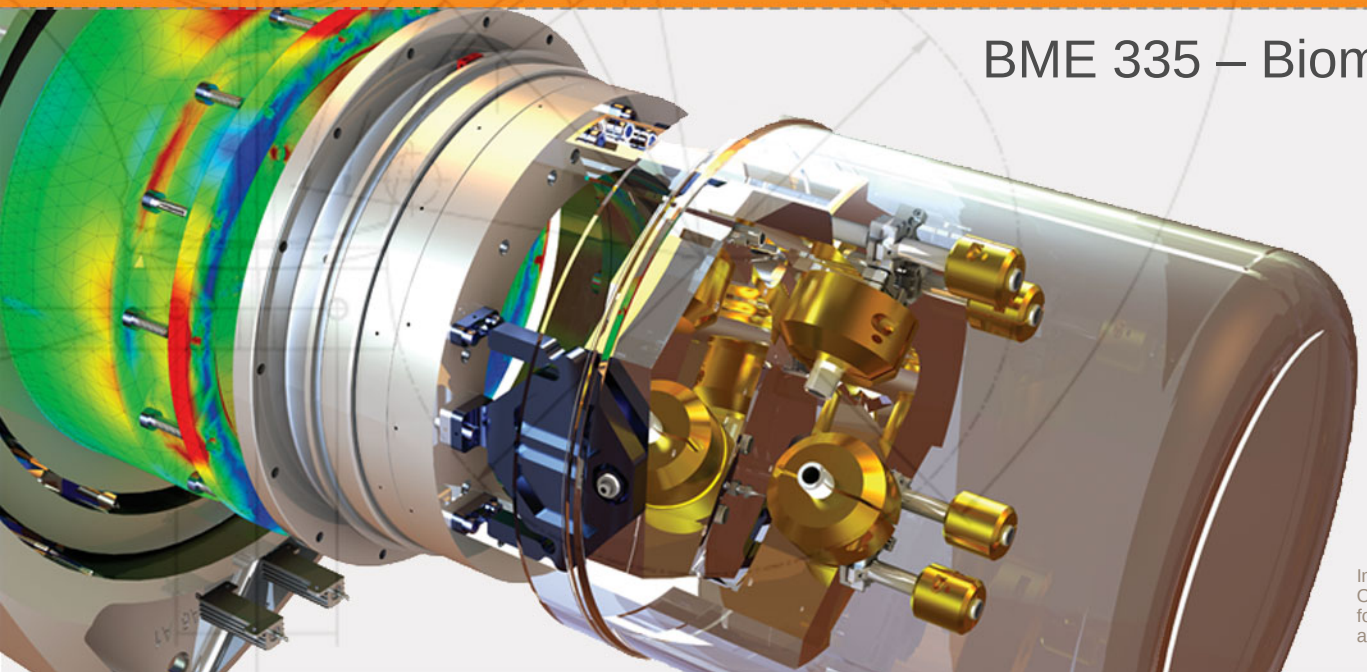
Wichita State University
College of Engineering
Biomedical Engineering

Matlab Lesson 1



BME 335 – Biomed Computer Apps

Image courtesy of National Optical Astronomy Observatory, operated by the Association of Universities for Research in Astronomy, under cooperative agreement with the National Science Foundation.

What is MATLAB®?

MATLAB® /Simulink® is a powerful software tool for:

- Performing mathematical computations and signal processing
- Analyzing and visualizing data (excellent graphics tools)
- Modeling physical systems and phenomena
- Testing engineering designs

Industry Applications

- **Biotech, Pharmaceutical, Medical:** Medical devices (e.g. hearing aids), imaging procedures, prosthesis, material analysis, computerized diagnostics…
- **Aircraft/Defense:** control and guidance system design and simulation, communications
- **Robotics:** design and control
- **Automotive:** cruise control, stability enhancement, fuel injection systems, hybrid power-train, sound suppression
- **Communications:** voice over internet, cell-phone, satellite, antenna design, wireless, error coding

Industry Applications (con't)

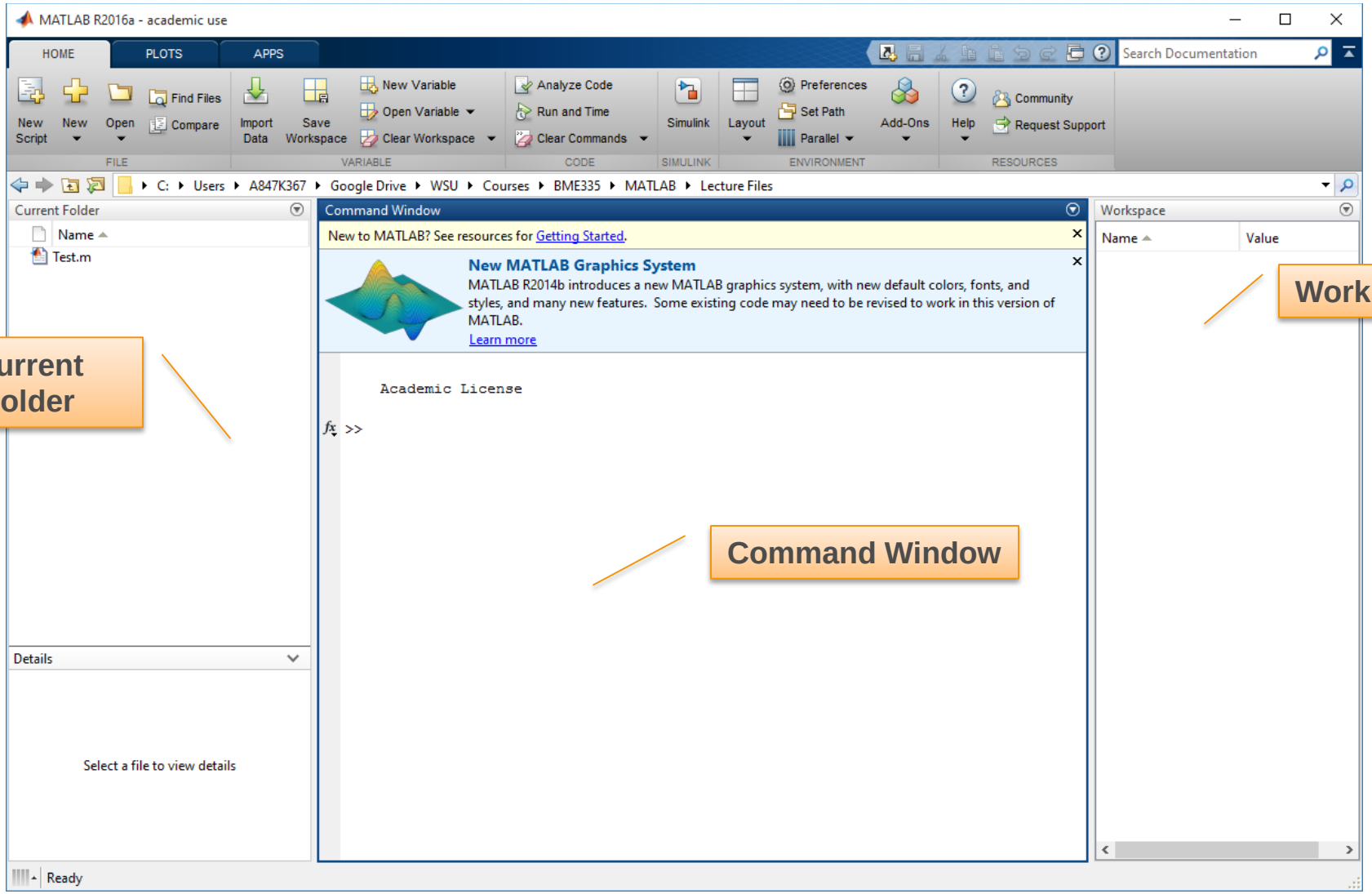
- **Electronics:** chip design, acoustics, voice processing and recognition
- **Industrial Automation and Machinery:** sensor design, machinery design and control
- **Utilities and Energy:** power conversion and control
- **Computers:** security systems, printer design
- **Financial:** portfolio management and risk, commodity trading, currency markets

MATLAB® BASICS

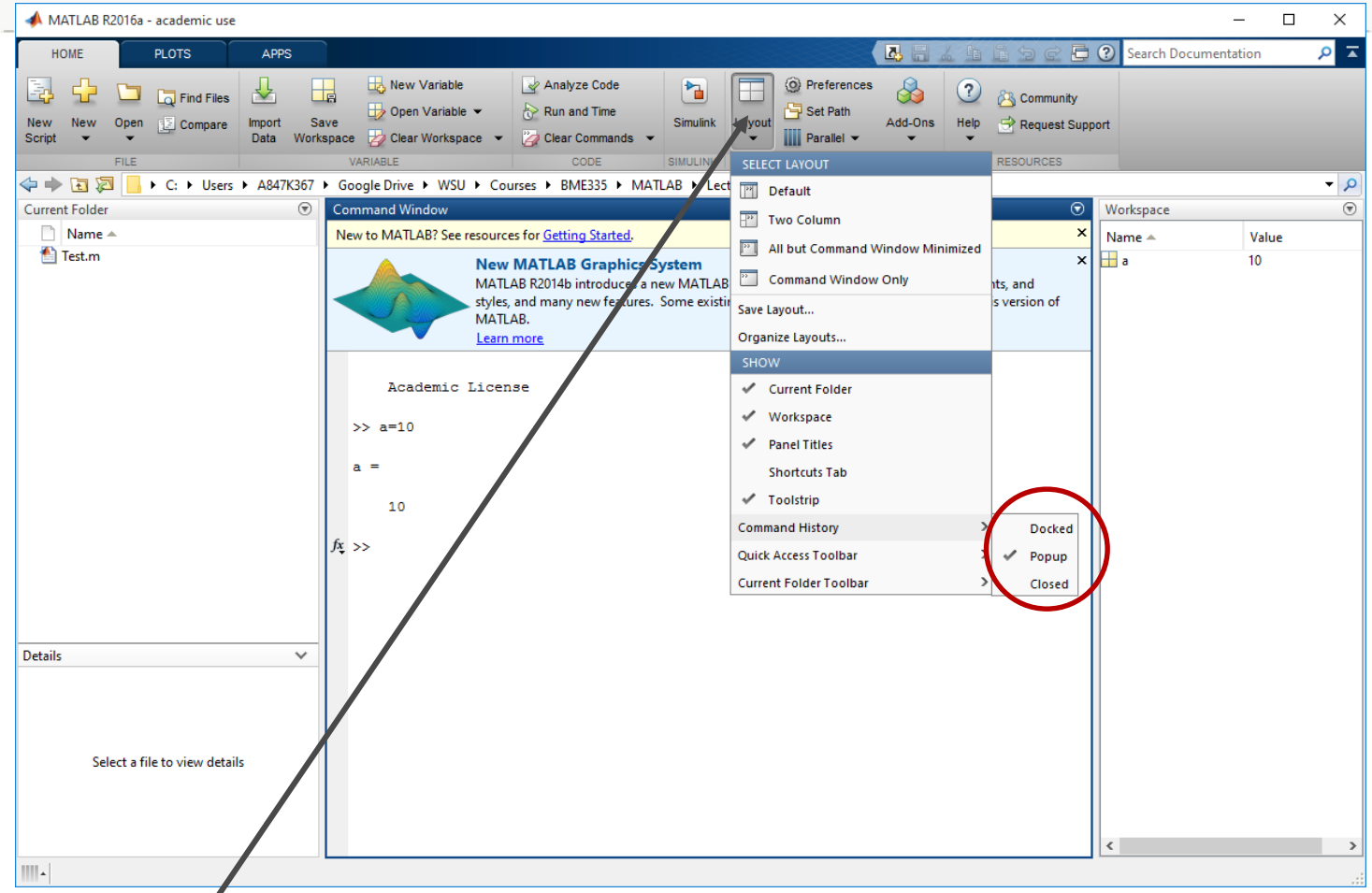
MATLAB Desktop

- The **Command window** is where you type MATLAB commands following the prompt: EDU>>
- The **Workspace window** shows all the variables you have defined in your current session. Variables can actually be manipulated within the workspace window.
- The **Current Folder** window displays all the files in whatever folder you select to be current.
- *(Not Default)* The **Command History** window displays all the MATLAB commands you have used recently – even includes some past sessions.

MATLAB Desktop

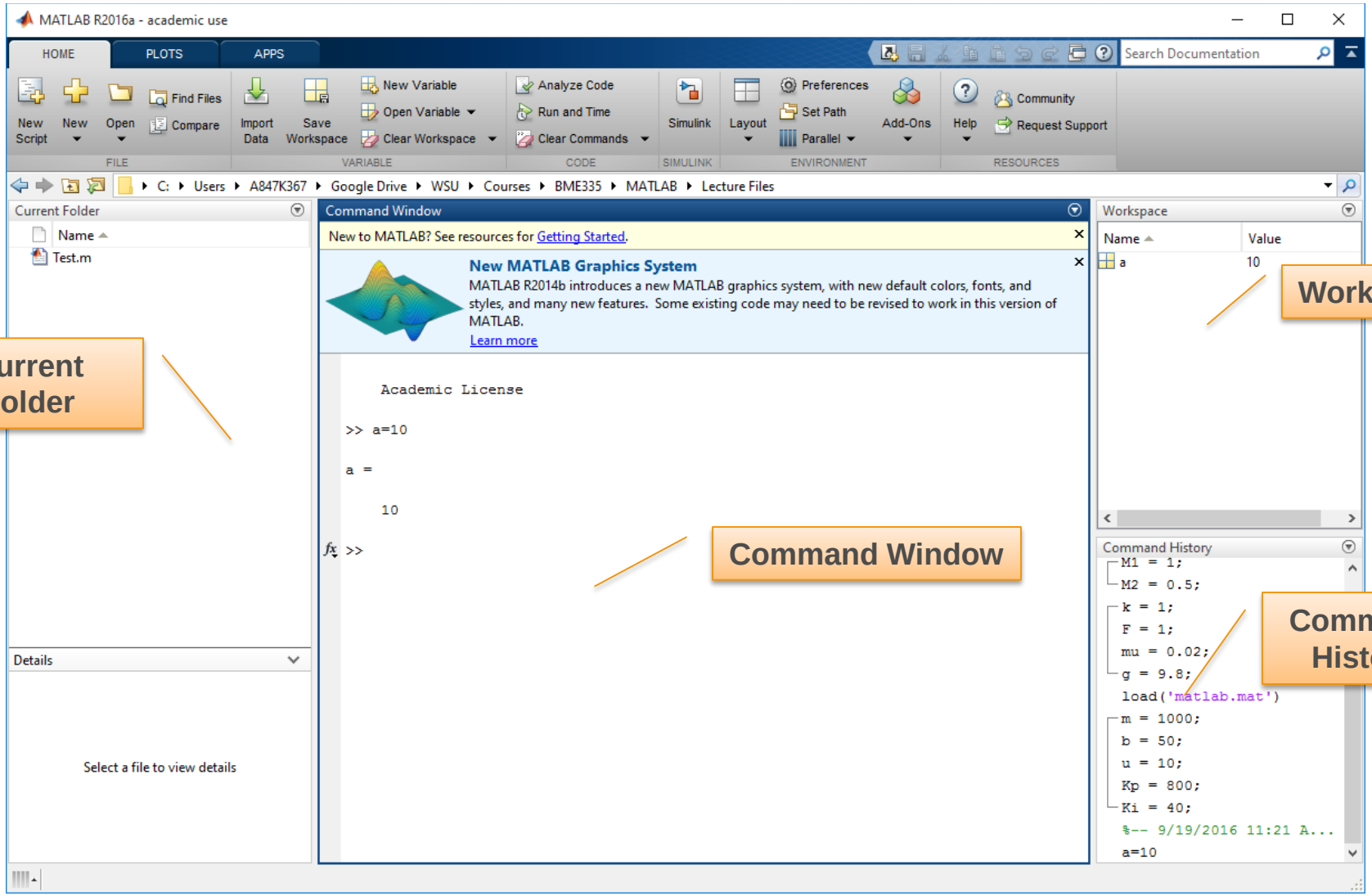


MATLAB Desktop



You can select what is on your desktop by Clicking on Layout. Go down to Command History and select docked.

MATLAB Desktop



Naming Rules for Variables

1. Variable names must begin with a letter
2. Names can include any combinations of letters, numbers, and underscores
3. Maximum length for a variable name is 63 characters
4. MATLAB® is case sensitive. The variable name A is different than the variable name a.
5. Avoid the following names: i, j, pi, and all built-in MATLAB® function names such as length, char, size, plot, break, cos, log, …
6. It is good programming practice to name your variables to reflect their function in a program rather than using generic x, y, z variables.

Creating Variables & Assigning Values

At the MATLAB command prompt (>>) type:

```
x = 10.57;
```

What happens? (Hint: look at workspace window)

Several things happen with this simple MATLAB command:

A variable, **x**, of type double is created
A memory location for the variable **x** is assigned
The value **10.57** is stored in that memory location called **x**.

Creating Variables & Assigning Values

At the MATLAB command prompt (>>) type:

```
x = 73.65
```

What happens?

The old value for **x** (10.57) is replaced by the new value (**73.65**)

Also, since the semicolon was left off the end, we see the result in the command window (as well as in the workspace window)

Creating Variables & Assigning Values

Variable do not have to be numbers. At the MATLAB command prompt type:

```
month = 'August'
```

What happens?

Several things happen with this simple MATLAB command:

- A variable, **month**, of type string (character array) is created
- A memory location for the variable **month** is assigned
- The string **August** is stored in that memory location called **month**.
- Notice that to enter a string, we must put single quotes around it

Displaying Variables

We can display a variable (i.e., show its value) by simply typing the name of the variable at the command prompt (leaving off the semicolon). Try this!

We can also use a function called **`disp`** to display variables.

Type the following commands at the command prompt:

```
>>  disp('The value of x is:'); disp(x)
```

Arithmetic Operators and Order of Operations

- Addition (+), Subtraction (-), Multiplication (*), Division (/), Power (^)
- Order of Operations (same math rules apply)
    1. Complete all calculations inside parenthesis or brackets using the precedent rules below
    2. Powers (left to right)
    3. Multiplication and Division (left to right)
    4. Addition and Subtraction (left to right)

Arithmetic Operators and Order of Operations

Some Examples:

```
>>  10/5*2

>>  5*2^3+4(2)

>> -1^4

>>  8^1/3
```

Exercise 1

1. In MATLAB create two variables: a = 4 and b = 17.2
2. Now use MATLAB to perform the following set of calculations:

$$5a$$

$$\sqrt[3]{b + 9.8}$$

$$10\sqrt{5a + 16}$$

$$c = \sqrt{a^2 + b^2}$$

Some MATLAB® Functions

Function	MATLAB®	Function	MATLAB®
cosine	cos or cosd	square root	sqrt
sine	sin or sind	exponential	exp
tangent	tan or tand	logarithm (base 10)	log10
cotangent	cot or cotd	natural log (base e)	log
arc cosine	acos or acosd	round to nearest integer	round
arc sine	asin or asind	round down to integer	floor
arc tangent	atan or atand	round up to integer	ceil
arc cotangent	acot or acotd		

Note: cos(α) assumes α in radians; whereas, cosd(α) assumes α in degrees. acos(x) returns the angle in radians; whereas, acosd(x) returns the angle in degrees.

π radians = 180 degrees

Other Useful Stuff

- `clear` clears all variables in the MATLAB® workspace
- `clear a, b` just clears variables a and b
- `clc` clears the command window
- `i` and `j` are defined in MATLAB to be $\sqrt{-1}$. If you define these variables to be something else, you lose the ability to work with complex numbers. So, avoid using i and j as variables.
- `pi` is defined in MATLAB as `3.14159`….

Help!

- The `help` command provides information about a function. Type `help cos` at the command prompt. This only works if you know the name of the function you want help with.

- You can also click on Help in the MATLAB toolbar and search for information by keyword(s).

STORING VARIABLES

Storing Variables

Suppose we type the following commands in MATLAB:

```
>>  y = 42;
>>  FavoriteDay = ‘Friday’
```

We know MATLAB stores the values associated with the variables, y and FavoriteDay, in memory.

How are these values stored?

Storing Variables

Computers store all data (numbers, letters, instructions, …) as strings of 1s and 0s (bits).

A **bit** is short for **bi**nary digi**t.** Like a switch, it has only two possible values:

On (1) or Off (0).

Everything (numbers, characters, images ..) are represented in a computer by 1s and 0s

Binary Number System

- The binary number system is a base 2 number system
- (0 or 1). It is based on powers of 2.
- The decimal number system is a base 10 number system
- (0, 1, 2, … 9). It is based on powers of 10.
- What does 5312 mean in a base 10 system?

5 3 1 2 = 5*10^3 + 3*10^2 + 1*10^1 + 2*10^0

1000	100	10	1
10^3	10^2	10^1	10^0

Binary Number System

- So what do the following binary numbers translate to in a decimal system?

0 0 1 0 0 0 1 1

2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0

1*2^5 + 1*2^1 + 1*2^0 = 35

1 0 0 1 0 0 0 1

2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0

1*2^7 + 1*2^4 + 1*2^0 = 145

1 0 1 0 1 0 1 0

128	64	32	16	8	4	2	1

1*128 + 1*32 + 1*8 + 1*2 = 170

Terminology

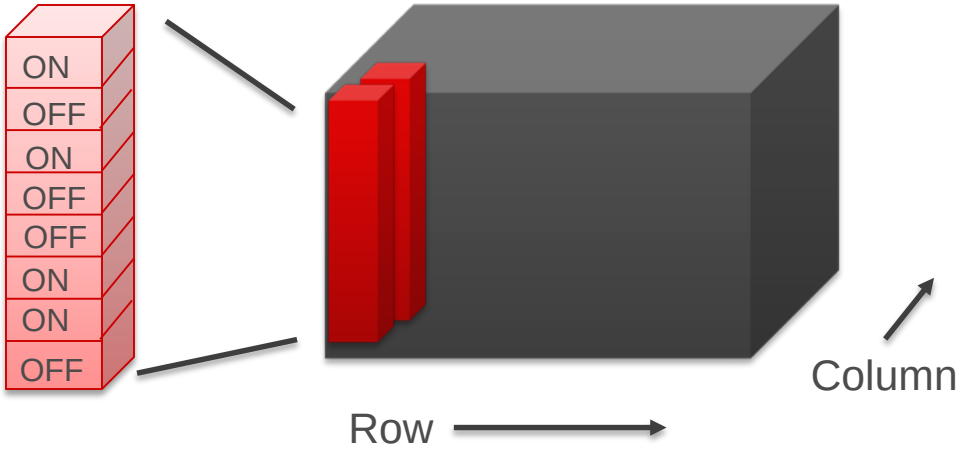
A **byte** is simply a string of 8 bits.

A **kilobyte** (kB) is 1000 bytes (*actually 1024!)*

A **megabyte** (MB) is 1,000,000 bytes

A **gigabyte** (GB) is 1,000,000,000 bytes

These are stored in 2D array of storage “units”



Numeric Data Types

MATLAB has several different options for storing numbers as bits. Unless you specify otherwise, all numbers in MATLAB are stored as doubles.

Name	Description	Range
double	64 bit floating point	-1.79769313486232E308 to -4.94065645841247E-324 4.94065645841247E-324 to 1.79769313486232E308
single	32 bit floating point	–3.402823E38 to –1.401298E-45 1.401298E-45 to 3.402823E38
uint8	8 bit unsigned integer	Integers from 0 to 255
int8	8 bit signed integer	Integers from –128 to 127
uint16	16 bit unsigned integer	Integers from 0 to 65535
int16	16 bit signed integer	Integers from –32768 to 32767
uint32	32 bit unsigned integer	Integers from 0 to 4294967295
int32	32 bit signed integer	Integers from –2147483648 to 2147483647

Data Types:

int8, uint8, int16, uint16, int32, uint32

- These data types work for integers as long as the integers don't exceed the range for the data type chosen.
- They take up less memory space than doubles.
- They don't work for non-integers. If you create a variable that is an int8 and try to assign it a value of 14.8, that variable will be assigned a value of 15 instead (closest integer within the range).
- One common application for integer data types is image data (jpeg, png, …)

Data Types: double and single

- A double uses 64 bits to store a number.
- A single uses 32 bits to store a number.
- Doubles and singles can be used to represent both integers and non-integers.

Why should I care how data is stored?

Example: Perform each of the following calculations in your head.

a = 4/3

b = a – 1

c = 3*b

e = 1 – c

What does MATLAB get?

Why should I care how data is stored in a computer?

What does MATLAB get?

a = 4/3 = 1.3333

b = a – 1 = 0.3333

c = 3*b = 1.0000

e = 1 – c = 2.2204e-016

It is not possible to perfectly represent all real numbers using a finite string of 1s and 0s.

Comments

- Not all numbers can be represented exactly even using 64 bit doubles. If we do many, many calculations with numbers which are just a tiny bit off, that error can grow very, very large depending on the type of computations being performed.
- 64 bit doubles have a huge, but still limited range. What happens if we exceed it? Try the following:

```
>>  a = 373^1500
>>  b = factorial(172)
```

ASCII Code

When you press a key on your computer keyboard, the key that you press is represented by a letter or character and is translated to a binary code.

A = 1000001 (Decimal = 65)

a = 1100001 (Decimal = 97)

0 = 0110000 (Decimal = 48)

ASCII Code

Dec	Hex	Char	Dec	Hex	Char	Dec	Hex	Char	Dec	Hex	Char
0	00	Null	32	20	Space	64	40	@	96	60	`
1	01	Start of heading	33	21	!	65	41	A	97	61	a
2	02	Start of text	34	22	"	66	42	B	98	62	b
3	03	End of text	35	23	#	67	43	C	99	63	c
4	04	End of transmit	36	24	$	68	44	D	100	64	d
5	05	Enquiry	37	25	%	69	45	E	101	65	e
6	06	Acknowledge	38	26	&	70	46	F	102	66	f
7	07	Audible bell	39	27	'	71	47	G	103	67	g
8	08	Backspace	40	28	(	72	48	H	104	68	h
9	09	Horizontal tab	41	29	)	73	49	I	105	69	i
10	0A	Line feed	42	2A	*	74	4A	J	106	6A	j
11	0B	Vertical tab	43	2B	+	75	4B	K	107	6B	k
12	0C	Form feed	44	2C	,	76	4C	L	108	6C	l
13	0D	Carriage return	45	2D	-	77	4D	M	109	6D	m
14	0E	Shift out	46	2E	.	78	4E	N	110	6E	n
15	0F	Shift in	47	2F	/	79	4F	O	111	6F	o
16	10	Data link escape	48	30	0	80	50	P	112	70	p
17	11	Device control 1	49	31	1	81	51	Q	113	71	q
18	12	Device control 2	50	32	2	82	52	R	114	72	r
19	13	Device control 3	51	33	3	83	53	S	115	73	s
20	14	Device control 4	52	34	4	84	54	T	116	74	t
21	15	Neg. acknowledge	53	35	5	85	55	U	117	75	u
22	16	Synchronous idle	54	36	6	86	56	V	118	76	v
23	17	End trans. block	55	37	7	87	57	W	119	77	w
24	18	Cancel	56	38	8	88	58	X	120	78	x
25	19	End of medium	57	39	9	89	59	Y	121	79	y
26	1A	Substitution	58	3A	:	90	5A	Z	122	7A	z
27	1B	Escape	59	3B	;	91	5B	[	123	7B	{
28	1C	File separator	60	3C	<	92	5C	\	124	7C	|
29	1D	Group separator	61	3D	=	93	5D	]	125	7D	}
30	1E	Record separator	62	3E	>	94	5E	^	126	7E	~
31	1F	Unit separator	63	3F	?	95	5F	_	127	7F	□

ASCII Code

Example: If you were typing a document, the word:

‘Hello’

In Dec would translate to: 72 101 108 108 111

In Hex would translate to: 48 65 6C 6C 6F

Of course, it is actually stored in binary but a big long string of 1s and 0s is pretty hard to read!

Strings in MATLAB

- MATLAB stores strings as an array of characters using the ASCII code.
- Each letter in a string takes up two bytes (16 bits) and the two bytes are the binary representation of the decimal number listed in the ASCII table.

Try the following in MATLAB:

```
>> month = ‘August’
>> double(month)
```

SCRIPT FILES

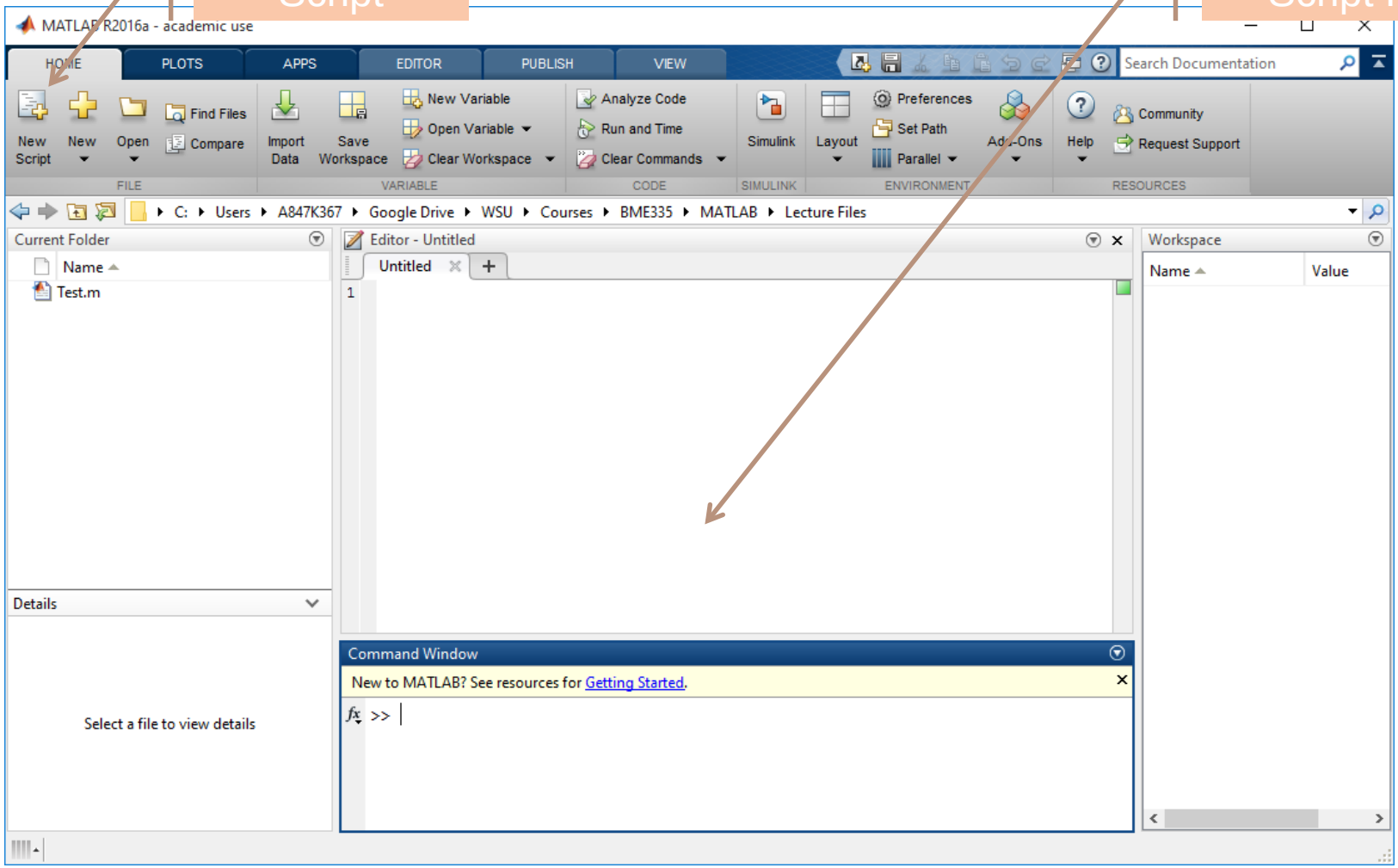
Script Files

- All of the pre-built commands that you use in MATLAB® are ***script files*** or ***functions*** (plot, mean, std, exp, cosd, …)
- MATLAB® allows the user to create his/her own customized m-files for specific applications or problems. (m-files are not too different from C-code)
- A script file is simply a collection of executable MATLAB® commands. To create a new script file, click on the New Script icon on the left side of the Home Tab.
- Function files will be covered in Models II.

Script Files

Click on New Script

Creates Blank Script File



Script File: Procedure

1. Type a set of executable commands in the editor window.

2. Save the file in an appropriate folder. ***When you pick a name for the file you must follow the same rules that MATLAB has for naming variables.***

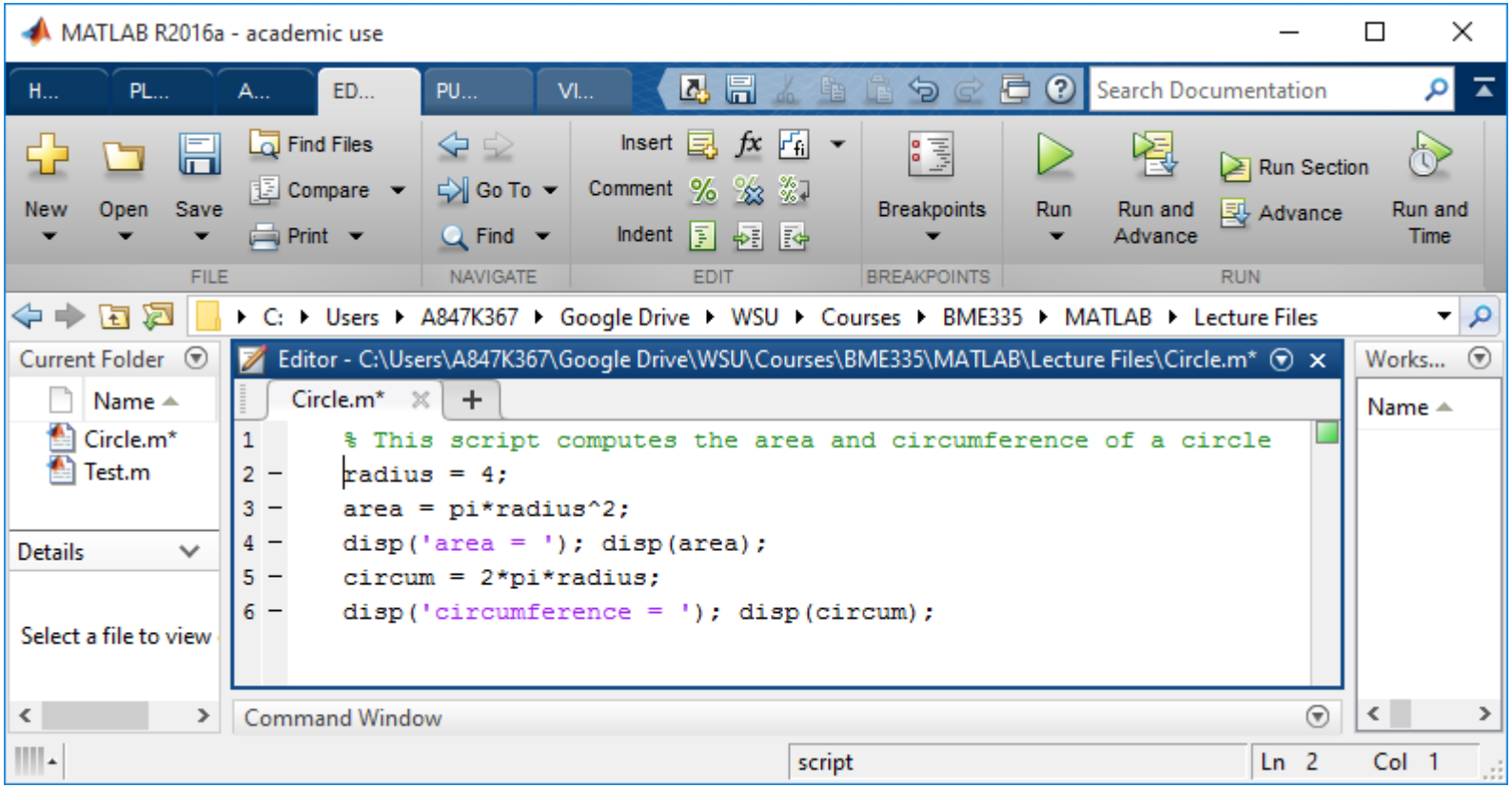
3. Set the current directory in MATLAB® to the same place where you saved the Green Run Arrow in the Editor window or simply type the name of the file (without the .m extension) at the command prompt in the MATLAB command window.

The file must be in your current directory or it will not run!

Exercise 2: New Script File

- Files should be saved on your Flash stick.
- Double click on the folder to make it your current folder.
- Clear your MATLAB workspace by typing `clear` at the command prompt.
- Click on New Script to open a blank script file.
- Type the commands on the next slide into the editor window then save the file as CircleScript in your newly created folder.

Exercise 2: Script File



Save the file as CircleScript on your flash stick.

Note: Any line that starts with a % is a comment and turns green – it doesn't execute.

Exercise 2: Run the Script File

- Now run your script file by clicking on the Green Arrow.
- Notice that every single variable defined in the script file (radius, area, and circum) appears in the Workspace Window. Area and circum are also displayed in the Command Window because of the **`disp`** command.
- Clear the workspace window by typing **`clear`** at the command prompt.
- At the command prompt, type the name of your script file: >> **`CircleScript`**. Note, that the results are exactly the same as before.

Script Files

- A script file is simply a set of executable MATLAB commands saved together in a file. It behaves exactly the same as entering each command sequentially in the command window. Every single variable defined appears in the workspace.
- Script files can be really useful if you are going to execute a lot of MATLAB commands. For example, suppose you execute 15 commands at the command prompt and discover an error in the first command that affected the last 14 commands. In the command window, you would have to fix the error in the first command then run the other 14 commands over again. If these commands were in a script file, you could fix the first command and re-run the script file – much faster !!
- They are also useful for repeated tasks