MAT 1206 – Introduction to MATLAB

CHAPTER 02: Fundamental Operators and Commands

Lesson 2

Content

- Variables
- Data types in MATLAB
- M-files

Variables

- In other programming languages, before use a variable must be defined. But MATLAB created it automatically and that make MATLAB easy to use.

```
x = 1+2
```

- Variables are used to store information to be referenced and manipulated in a computer programing.
- In computer programming, a variable is an abstract storage location paired with an associated symbolic **name**, which contains some known or unknown quantity of information referred to as a **value**.
- Assignment statement: <variable> = <expression>

Cont.

- In MATLAB environment, every variable is an array or matrix.

```
x = 3 % defining x and initializing it with a value
```

MATLAB will execute the above statement and return the following result:

```
x =

      3
```

It creates a 1-by-1 matrix named *x* and stores the value 3 in its element.

Cont.

- Another example:

```
x = sqrt(16) % defining x and initializing it with an expression
```

MATLAB will execute the above statement and return the following result:

```
x =

     4
```

Cont.

- Now type x = 3. If we enter x, MATLAB just search its memory for the variable name x and print the value.

```
>> x=3
x =
     3
```

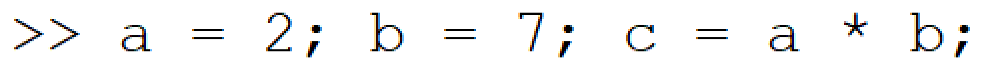
- Now try y.

```
>> y
Unrecognized function or variable 'y'.
```

Cont.

- Multiple Assignments:

  You can have multiple assignments on the same line.



```
>> a = 2; b = 7; c = a * b;
```

Variable Naming in MATLAB

- A MATLAB variable can only begin with letter followed by underscore _ and numbers inside or at the end of the variable name.
- MATLAB is case sensitive, so A and a are not the same variable.
- Other symbols are syntactically invalid anywhere in a variable name.
- Examples of valid names:

```
 x1
firstValue
first_value
```

Cont.

- Examples of invalid names:

  2x

  x*

  y!

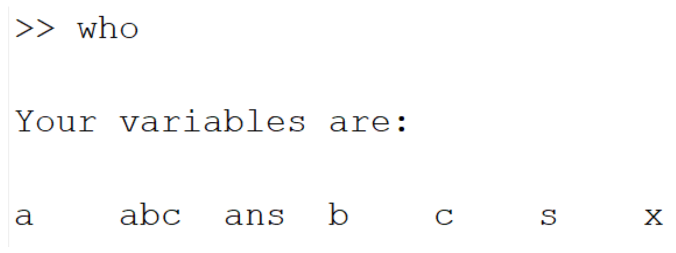
MATLAB reserved names (keywords)

- Variable names in MATLAB cannot be the same as MATLAB keywords, which are simply names that reserved for a specific purpose in MATLAB programming.
- They are used to define the syntax and structure of the MATLAB language. You cannot define variables with the exact same names as MATLAB keywords, such as ‘if’ or ‘end’.
- For a list of MATLAB keyword, run the ‘iskeyword’ command.
- For determine whether input is MATLAB keyword ‘iskeyword(‘end’)’.

Cont.

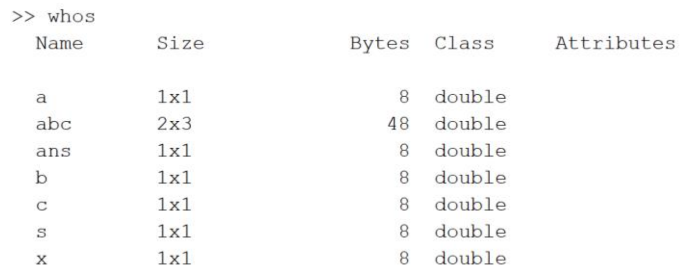
- If you have forgotten the Variables:

  The ‘who’ command displays all the variable names you have used.



```
>> who

Your variables are:

a     abc   ans   b     c     s     x
```

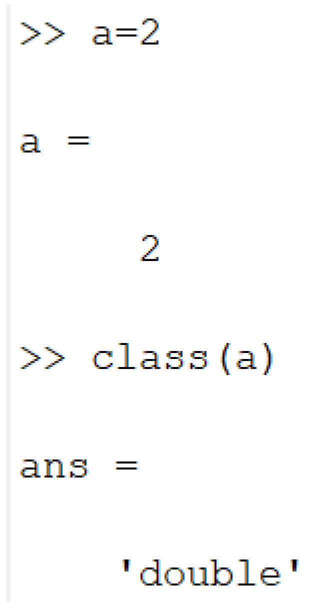
  The ‘whos’ command displays little more about the variables:



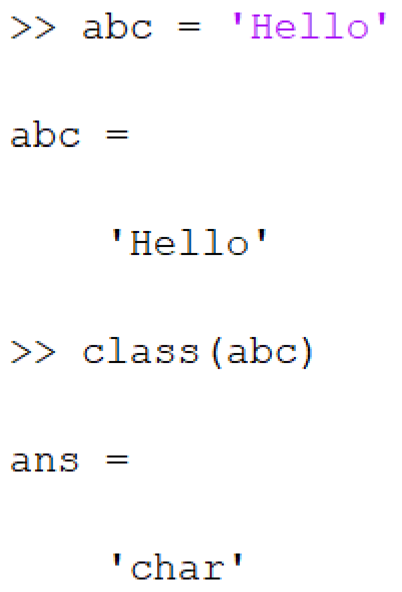
```
>> whos
  Name      Size            Bytes  Class     Attributes

  a         1x1                 8  double
  abc       2x3                48  double
  ans       1x1                 8  double
  b         1x1                 8  double
  c         1x1                 8  double
  s         1x1                 8  double
  x         1x1                 8  double
```

Cont.

The ‘class()’ command displays the type of a value/variable



```
>> a=2

a =

     2

>> class(a)

ans =

    'double'
```



```
>> abc = 'Hello'

abc =

    'Hello'

>> class(abc)

ans =

    'char'
```

Exercise

Define variables with the assignments $x = 5$, $y = 2.5$ and $X = 1/7$. Calculate the following within MATLAB.

(a) $\dfrac{5(y - x)}{14X - 19}$

(b) $\dfrac{9\sqrt{X}}{11}$

(c) $2 \sin x \, \sec y$

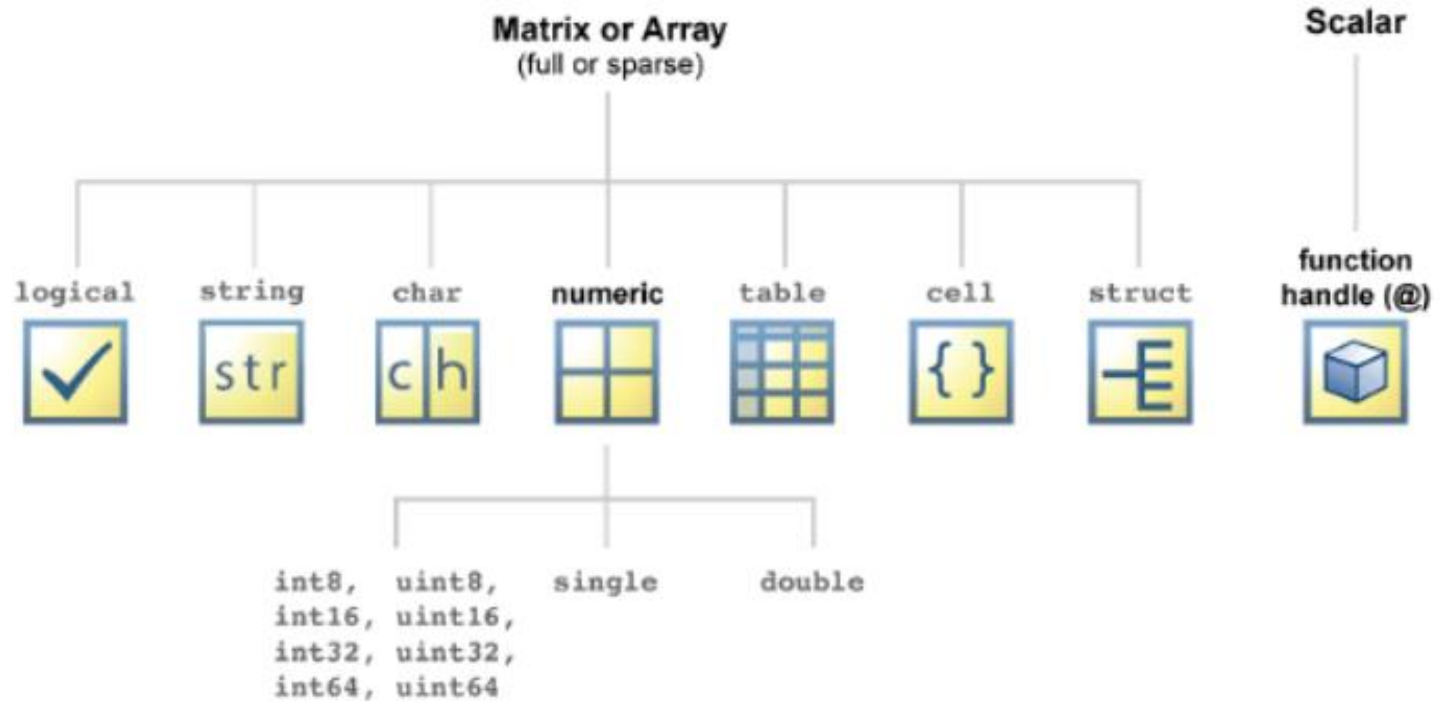
(d) $e^{(X+y)/x} + 6 \sqrt[3]{x}$

Data types in MATLAB

- There are many different data types, or classes, that you can work with in MATLAB.
- You can build matrices and arrays of floating-point and integer data, characters and strings, logical true and false values, and so on.
- Function handles connect your code with any MATLAB function regardless of the current scope.
- Tables, structures, and cell arrays provide a way to store dissimilar types of data in the same container.

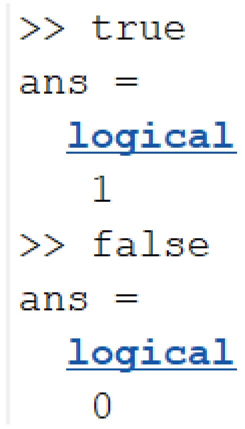
Cont.

- There are 17 fundamental classes in MATLAB.



Cont.

- Logical:
  - MATLAB represents Boolean data using the logical data type.
  - This data type represents true and false states using the numbers 1 and 0, respectively.
  - Certain MATLAB functions and operators return logical values to indicate fulfillment of a condition.

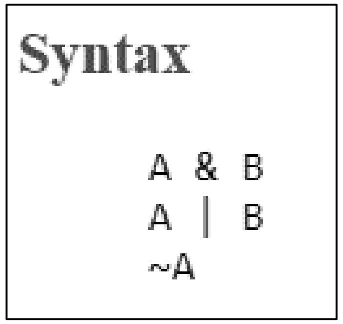


```
>> true
ans =
  logical
   1
>> false
ans =
  logical
   0
```

Cont.

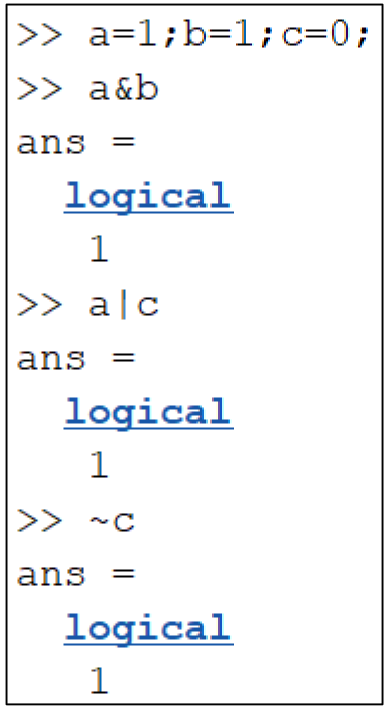
- Logical Operators, Element-wise & | ~

  The symbols &, |, and ~ are the logical array operators AND, OR, and NOT.



Syntax

```
A & B
A | B
~A
```



```
>> a=1;b=1;c=0;
>> a&b
ans =
  logical
   1
>> a|c
ans =
  logical
   1
>> ~c
ans =
  logical
   1
```

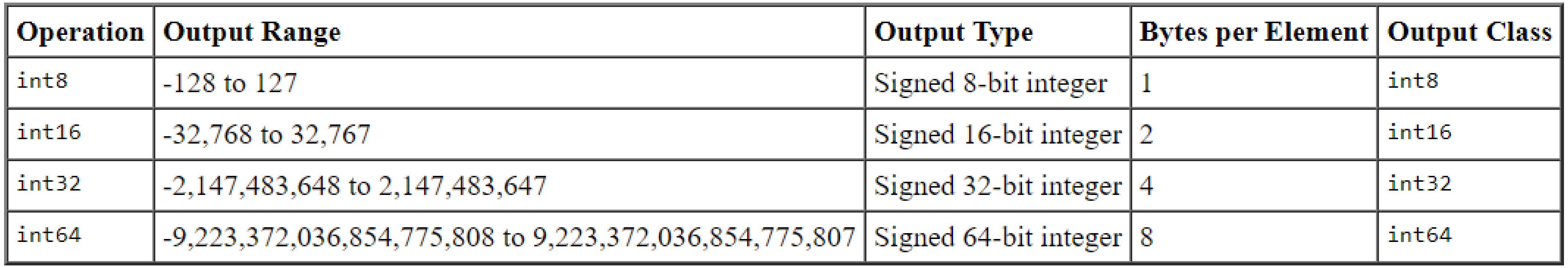
Cont.

- Characters (char) and Strings:
  - Character arrays and string arrays provide storage for text data in MATLAB.
  - A character array is a sequence of characters, just as a numeric array is a sequence of numbers. A typical use is to store short pieces of text as character vectors, such as c = 'Hello World'.
  - A string array is a container for pieces of text. String arrays provide a set of functions for working with text as data. You can create strings using double quotes, such as str = "Greetings friend".

Cont.

- Numeric:
  - int8, int16, int32, int64: Convert to signed integer

    Syntax: i = int8(x), i = int16(x), i = int32(x), i = int64(x)



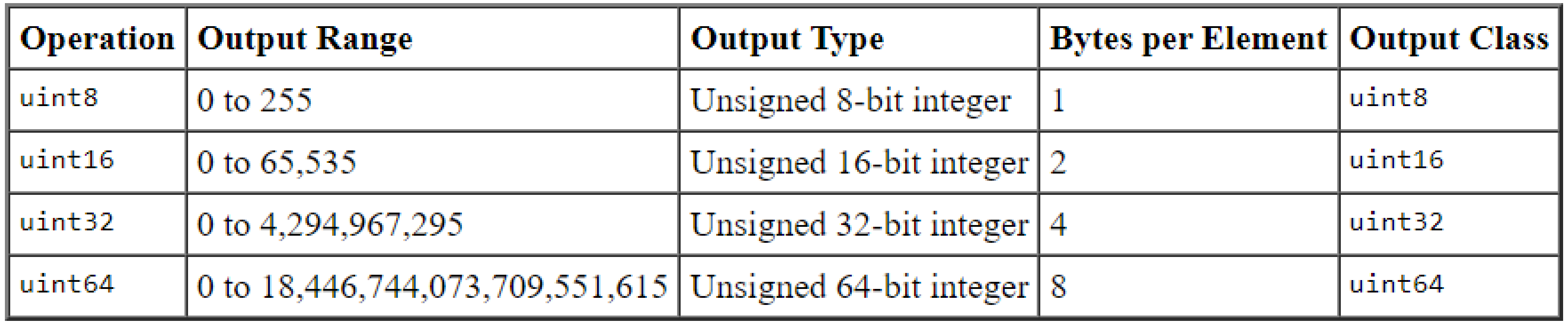
Operation	Output Range	Output Type	Bytes per Element	Output Class
`int8`	-128 to 127	Signed 8-bit integer	1	`int8`
`int16`	-32,768 to 32,767	Signed 16-bit integer	2	`int16`
`int32`	-2,147,483,648 to 2,147,483,647	Signed 32-bit integer	4	`int32`
`int64`	-9,223,372,036,854,775,808 to 9,223,372,036,854,775,807	Signed 64-bit integer	8	`int64`

A value of x above or below the range for a class is mapped to one of the endpoints of the range.

Cont.

- uint8, uint16, uint32, uint64: Convert to unsigned integer

Syntax: i = uint8(x), i = uint16(x), i = uint32(x), i = uint64(x)



Operation	Output Range	Output Type	Bytes per Element	Output Class
`uint8`	0 to 255	Unsigned 8-bit integer	1	`uint8`
`uint16`	0 to 65,535	Unsigned 16-bit integer	2	`uint16`
`uint32`	0 to 4,294,967,295	Unsigned 32-bit integer	4	`uint32`
`uint64`	0 to 18,446,744,073,709,551,615	Unsigned 64-bit integer	8	`uint64`

A value of x above or below the range for a class is mapped to one of the endpoints of the range.

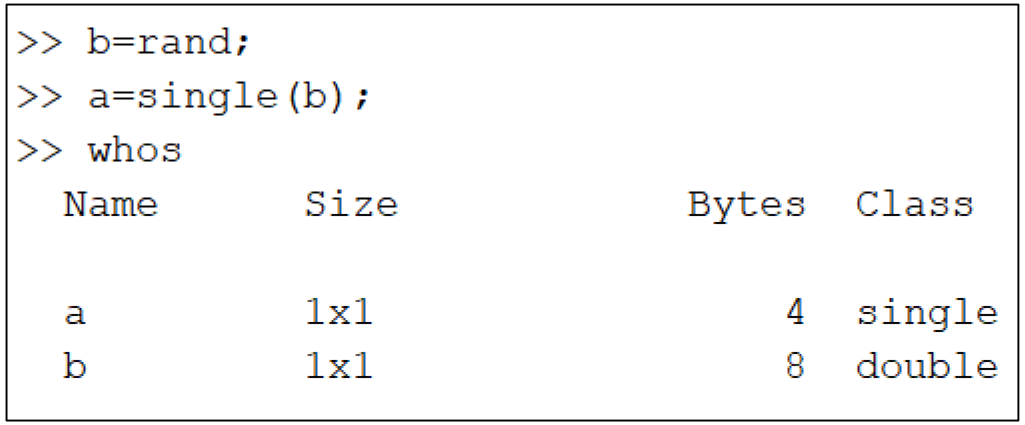
Cont.

- single: Convert to single-precision

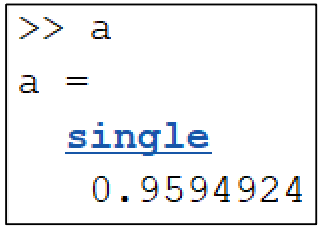
  Syntax: B = single(A)

- double: Convert to double-precision

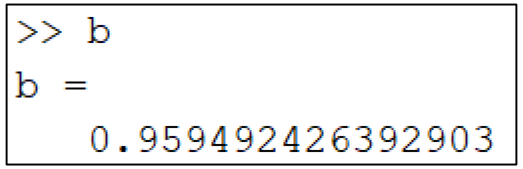
  Syntax: B = double(A)



```
>> b=rand;
>> a=single(b);
>> whos
  Name      Size            Bytes  Class

  a         1x1                 4  single
  b         1x1                 8  double
```



```
>> a
a =
  single
   0.9594924
```

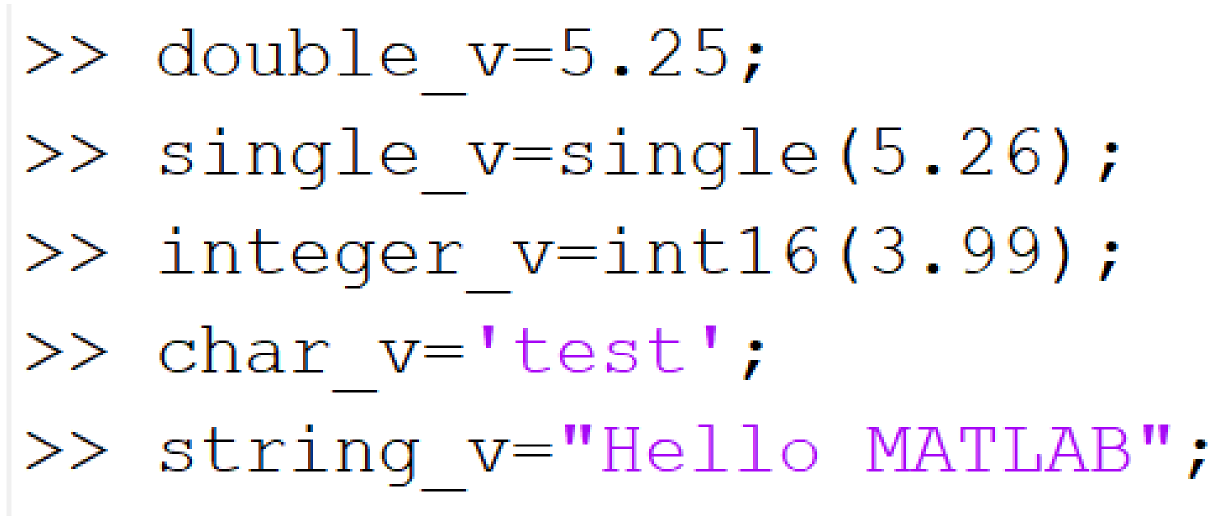


```
>> b
b =
   0.95949242639290
```

Because MATLAB stores numbers of type single using 32 bits, they require less memory than numbers of type double , which use 64 bits. However, because they are stored with fewer bits, numbers of type single are represented to less precision than numbers of type double.

Cont.

- For the moment, try the following data type:



```
>> double_v=5.25;
>> single_v=single(5.26);
>> integer_v=int16(3.99);
>> char_v='test';
>> string_v="Hello MATLAB";
```

Cont.

```
>> whos
  Name           Size             Bytes  Class     Attributes

  char_v         1x4                  8  char
  double_v       1x1                  8  double
  integer_v      1x1                  2  int16
  single_v       1x1                  4  single
  string_v       1x1                166  string
```

Introduction to M-files

- What is an m-file?
- Why use m-files?
- How to create, save or open an m-file?
- How to run the m-file?

Cont.

- What is an m-file?
  - An m-file, or script file, is a simple text file where you can place MATLAB commands. When the file is run, MATLAB reads the commands and executes them exactly as it would if you had typed each command sequentially at the MATLAB prompt.
  - All m-file names must end with the extension '.m' (e.g. test.m).

Cont.

- Why use m-files?
  - For simple problems, entering your requests at the MATLAB prompt is fast and efficient.
  - However, as the number of commands increases or trial and error is done by changing certain variables or values, typing the commands over and over at the MATLAB prompt becomes tedious.
  - M-files will be helpful and almost necessary in these cases.

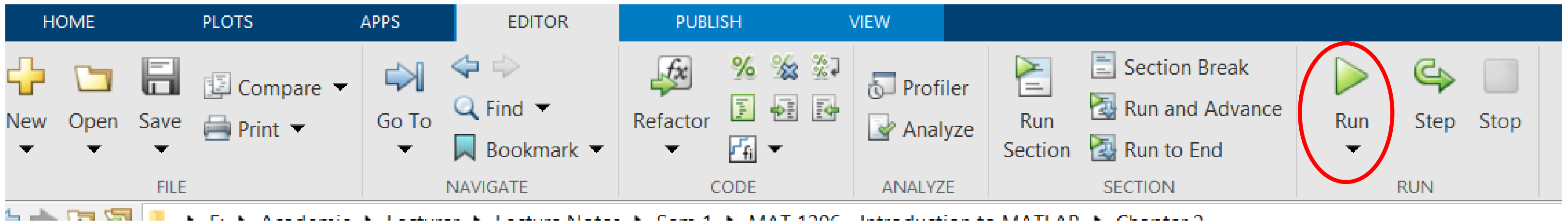
Cont.

- How to create, save or open an m-file?
  - To create an m-file, choose New from the File menu and select Script. This procedure brings up a text editor window in which you can enter MATLAB commands.
  - To save the m-file, simply go to the File menu and choose Save (remember to save it with the '.m' extension).
  - To open an existing m-file, go to the File menu and choose Open.

Cont.

- How to run the m-file?
  - After the m-file is saved with the name filename.m in the current MATLAB folder or directory, you can execute the commands in the m-file by simply typing filename at the MATLAB command window prompt.

    or
  - Use the Run button:



Saving Your Work (Data)

- The ‘save’ command is used for saving all the variables in the workspace, as a file with .mat extension, in the current directory.

  For example,

```
save myfile
```

- You can reload the file anytime later using the ‘load’ command.

  For example,

```
load myfile
```

Questions/queries?