EX. NO: 7

ENTERING COMMANDS, CREATING NUMERIC VARIABLES AND CREATING CHARACTER VARIABLES

Aim:

1. To learn and practice how to launch and use MATLAB IDE
2. To familiarize with various categories of MATLAB commands
3. To create numeric variables and character variables
4. To learn and use data types and operators

Procedure

MATLAB development IDE (Integrated Development Environment)can be launched from the icon created on the desktop. The main working window in MATLAB is called the desktop. When MATLAB is started, the desktop appears in its default layout:

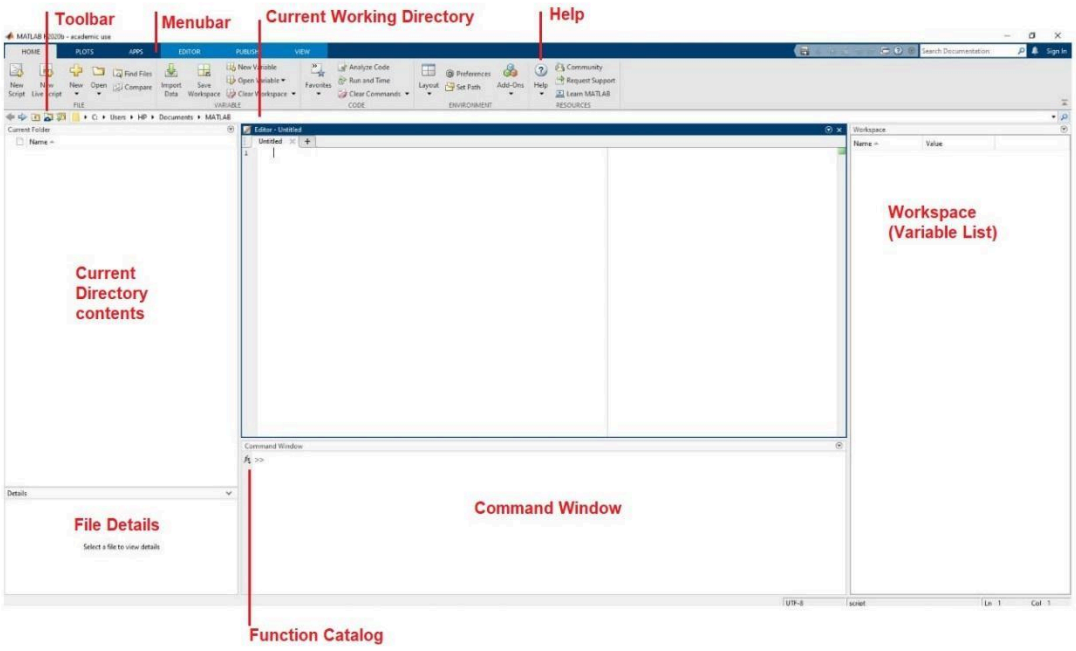

Figure 1: MATLAB Integrated Development Environment (IDE)

MATLAB provides a Desktop GUI based environment. The Home tab of the environment contains three panels there:

1. **Current folder:** It is located on the left side; here, you can access your files.
2. **Command Window:** It is located on the right side; it is the command prompt, and here you can enter commands to operate the functions, to assign variables, and for calculations.
3. **Workspace:** It is located on the left side right below the Current Folder; here, all variables that you create are stored, and data from other files can also be imported here.

Current Folder- This panel allows you to access the project folders and files.

Figure 2: Current Folder

Command Window - This is the main area where commands can be entered at the command line. It is indicated by the command prompt (>>).

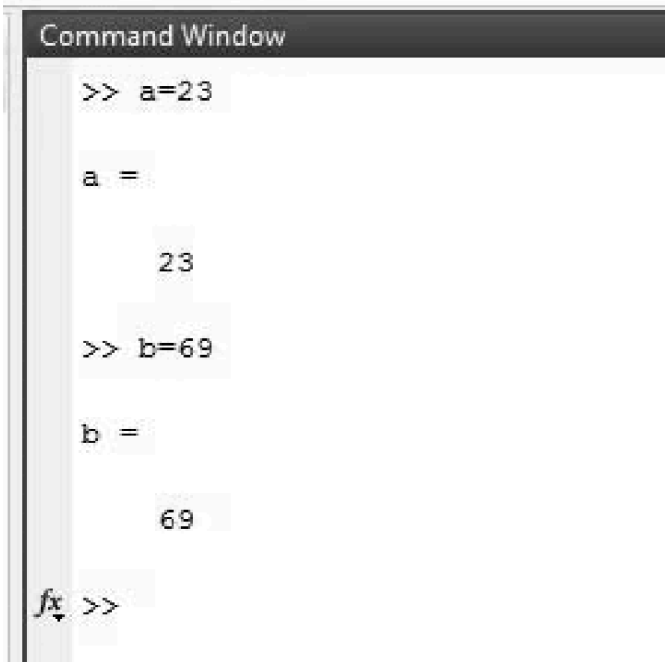

Figure 3: Command Window

Workspace - The workspace shows all the variables created and/or imported from files.

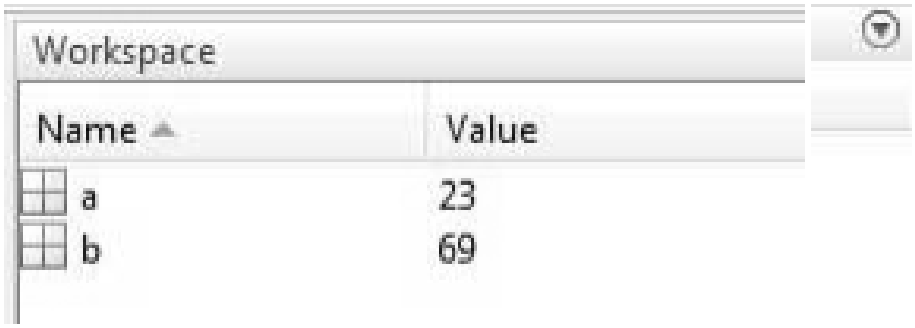

Figure 4: Workspace

Command History - This panel shows or rerun commands that are entered at the command line.

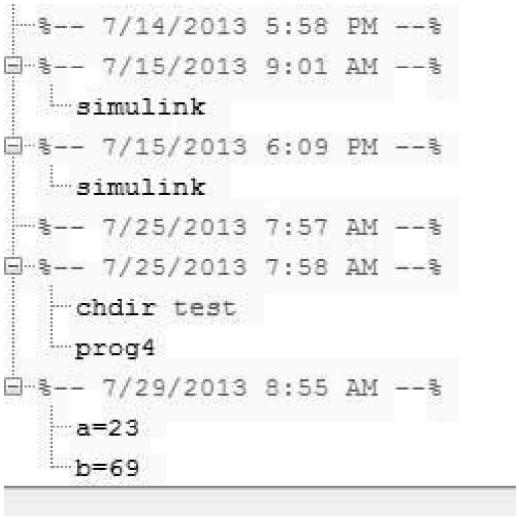

Figure 5: Command History

Creating MATLAB variables

Rules for variable names are:

1. They are case sensitive: nice, NICE,NiCe and NicE are all different.
2. They can have a length of up to 31 characters Pachycephalosaurus is a valid variable name (although it is quite frustrating to type).
3. They must start with a letter, followed by any combination of letters, numbers and underscores. For example, week1, week1_prac, MATLAB_prac_1_is_fun,x are all valid variable names.

4. There are some reserved words that can't be used as a variable name, i.e. help, pi, sin, pretty, etc.
5. clear statements are used to delete the values of previous variables. clear all deletes all values, whereas clear just deletes the variable you nominate.

MATLAB variables are created with an assignment statement. The syntax of variable assignment is

variable name = a value (or an expression)

For example,

>> x = expression

Where expression is a combination of numerical values, mathematical operators, variables, and function calls.

Overwriting variable

Overwriting variable Once a variable has been created, it can be reassigned. In addition, if you do not wish to see the intermediate results, you can suppress the numerical output by putting a semicolon (;) at the end of the line.

Then the sequence of commands looks like this:

>> t = 5;
>> t = t+1 t = 6 1.4.3

Error messages

Error messages If we enter an expression incorrectly, MATLAB will return an error message. For example, in the following, we left out the multiplication sign, *, in the following expression

```
>> x = 10;
>> 5x
??? 5x
     |
Error: Unexpected MATLAB expression.
```

Creating character variables

Creating a character string is quite simple in MATLAB. In fact, we have used it many times. For example, you type the following in the command prompt:

```
my_string = 'Hello World'
```

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

```
my_string = Hello World
```

MATLAB considers all variables as arrays, and strings are considered as character

arrays. Let us use the whos command to check the variable created above:

```
whos
```

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

Name	Size	Bytes Class	Attributes
my_string	1x16	32 char	

Interestingly, you can use numeric conversion functions like **uint8** or **uint16** to convert the characters in the string to their numeric codes. The **char** function converts the integer vector back to characters:

Example

Create a script file and type the following code into it:

```
my_string = 'Hello World';
str_ascii = uint8(my_string)          % 8-bit ascii values
str_back_to_char= char(str_ascii)
str_16bit = uint16(my_string)         % 16-bit ascii values
str_back_to_char = char(str_16bit)
```

When you run the file, it displays the following result:

```
str_ascii =
 Columns 1 through 14
```

84 117 116 111 114 105 97 108 39 115 32 80 111 105

Columns 15 through 16

110 116

str_back_to_char =

Hello World

str_16bit =

Columns 1 through 10

84 117 116 111 114 105 97 108 39 115

Columns 11 through 16

32 80 111 105 110 116

str_back_to_char =

Hello World

Rectangular Character Array

The strings we have discussed so far are one-dimensional character arrays; however, we need to store more than that. We need to store more dimensional textual data in our program. This is achieved by creating rectangular character arrays.

Simplest way of creating a rectangular character array is by concatenating two or more one-dimensional character arrays, either vertically or horizontally as required.

You can combine strings vertically in either of the following ways:

Using the MATLAB concatenation operator **[]** and separating each row with a semicolon (;).Please note that in this method each row must contain the same number of characters. For strings with different lengths, you should pad with space characters as needed.

Using the char function. If the strings are of different lengths, char pads the shorter strings with trailing blanks so that each row has the same number of characters.

Example

Create a script file and type the following code into it:

```
doc_profile = ['Zara Ali                                ']; ...
              'Sr. Surgeon                              '; ...
              'R N Tagore Cardiology Research Center']
doc_profile = char('Zara Ali', 'Sr. Surgeon', ...
                   'RN Tagore Cardiology Research Center')
```

When you run the file, it displays the following result:

```
doc_profile =
Zara Ali
Sr. Surgeon
R N Tagore Cardiology Research Center
doc_profile =
Zara Ali
Sr. Surgeon
RN Tagore Cardiology Research Center
```

You can combine strings horizontally in either of the following ways:

Using the MATLAB concatenation operator, [] and separating the input strings with a comma or a space. This method preserves any trailing spaces in the input arrays. Using the string concatenation function, **strcat**. This method removes trailing spaces in the inputs

Example

Create a script file and type the following code into it:

```
name = 'Zara Ali ';
position = 'Sr. Surgeon ';
worksAt = 'R N Tagore Cardiology Research Center';
profile = [name ', ' position ', ' worksAt]
profile = strcat(name, ', ', position, ', ', worksAt)
```

When you run the file, it displays the following result:

```
profile =
Zara Ali , Sr. Surgeon , R N
Tagore Cardiology Research Center
profile =
Zara Ali,Sr. Surgeon,R N Tagore Cardiology Research Center
```

Combining Strings into a Cell Array

From our previous discussion, it is clear that combining strings with different lengths could be a pain as all strings in the array has to be of the same length.
We have used blank spaces at the end of strings to equalize their length. However, a more efficient way to combine the strings is to convert the resulting array into a cell array.

MATLAB cell array can hold different sizes and types of data in an array. Cell arrays provide a more flexible way to store strings of varying length.

The ***cellstr*** function converts a character array into a cell array of strings.

Example

Create a script file and type the following code into it:

```
name = 'Zara Ali ';
position = 'Sr. Surgeon ';
worksAt = 'R N Tagore Cardiology Research Center';
profile = char(name, position, worksAt);
profile = cellstr(profile);
disp(profile)
```

When you run the file, it displays the following result:

```
'Zara Ali'
'Sr. Surgeon'
'R N Tagore Cardiology Research Center'
```

String Functions in MATLAB

MATLAB provides numerous string functions creating, combining, parsing, comparing and manipulating strings.

Following table provides brief description of the string functions in MATLAB:

Function	Purpose
Functions for storing text in character arrays, combine character arrays, etc.	
blanks	Create string of blank characters
cellstr	Create cell array of strings from character array
char	Convert to character array (string)
iscellstr	Determine whether input is cell array of strings
ischar	Determine whether item is character array
sprintf	Format data into string
strcat	Concatenate strings horizontally
strjoin	Join strings in cell array into single string
Functions for identifying parts of strings, find and replace substrings	
ischar	Determine whether item is character array
isletter	Array elements that are alphabetic letters
isspace	Array elements that are space characters
isstrprop	Determine whether string is of specified category
sscanf	Read formatted data from string
strfind	Find one string within another
strrep	Find and replace substring
strsplit	Split string at specified delimiter
strtok	Selected parts of string
validatestring	Check validity of text string
symvar	Determine symbolic variables in expression
regexp	Match regular expression (case sensitive)
regexpi	Match regular expression (case insensitive)
regexprep	Replace string using regular expression
regexptranslate	Translate string into regular expression
Functions for string comparison	
strcmp	Compare strings (case sensitive)
strcmpi	Compare strings (case insensitive)
strncmp	Compare first n characters of strings (case sensitive)
strncmpi	Compare first n characters of strings (case insensitive)
Functions for changing string to upper- or lowercase, creating or removing white space	
deblank	Strip trailing blanks from end of string
strtrim	Remove leading and trailing white space from string
lower	Convert string to lowercase
upper	Convert string to uppercase
strjust	Justify character array

Examples

The following examples illustrate some of the above-mentioned string functions:

Formatting Strings

Create a script file and type the following code into it:

```
A = pi*1000*ones(1,5);
sprintf(' %f \n %.2f \n %+.2f \n %12.2f \n %012.2f \n', A)
```

When you run the file, it displays the following result:

```
ans =
3141.592654
3141.59
+3141.59
 3141.59
000003141.59
```

Joining Strings

Create a script file and type the following code into it:

```
%cell array of strings
str_array = {'red','blue','green', 'yellow', 'orange'};
% Join strings in cell array into single string
str1 = strjoin("-", str_array)
str2 = strjoin(",", str_array)
```

When you run the file, it displays the following result:

```
str1 =
red blue green yellow orange
str2 =
red , blue , green , yellow , orange
```

Finding and Replacing Strings

Create a script file and type the following code into it:

```
students = {'Zara Ali', 'Neha Bhatnagar', ...
 'Monica Malik', 'Madhu Gautam', ...
 'Madhu Sharma', 'Bhawna Sharma',...
 'Nuha Ali', 'Reva Dutta', ...
 'Sunaina Ali', 'Sofia Kabir'};
% The strrep function searches and replaces sub-string.
new_student = strrep(students(8), 'Reva', 'Poulomi')
% Display first names
first_names = strtok(students)
```

When you run the file, it displays the following result:

```
new_student =
'Poulomi Dutta'
first_names =
 Columns 1 through 6
 'Zara' 'Neha' 'Monica' 'Madhu' 'Madhu' 'Bhawna'
 Columns 7 through 10
 'Nuha' 'Reva' 'Sunaina' 'Sofia'
```

Comparing Strings

Create a script file and type the following code into it:

```
str1 = 'This is test'
str2 = 'This is text'
if (strcmp(str1, str2))
sprintf('%s and %s are equal', str1, str2)
else
sprintf('%s and %s are not equal', str1, str2)
end
```

When you run the file, it displays the following result:

```
str1 =
This is test
str2 =
This is text
ans =
This is test and This is text are not equal
```

Work to be done:

1. A love-struck engineer specialising in planning the foundations of a building wrote the following code to calculate the area of a rectangle.

> Determine what each variable represents and then rename the variables to write the new code in MATLAB with meaningful variable names.

```
dinner = 4

nice = 7

roses = dinner*nice
```

2. Use MATLAB to evaluate the following functions at the given values of x . Type in the command clear x beforehand.

F = $\sqrt{3x - 6}$, $x = 5$ (answer is 3)

G = $\frac{x}{|x|}$, $x = 3$, $x = -3$ (answer are 1,-1)

H = $2x^2 - x$, $x = 2$ (answer are 6)