Practical work 03: Read, display and save data in MATLAB

Practical work 03: Read, display and save data in MATLAB

Reading, displaying and saving data in MATLAB are essential operations for data manipulation and analysis. Firstly, MATLAB environment offers several ways to read data stored in different file formats. After reading and obtaining data, the displaying operations are crucial for analysis and visualization. In this context, the MATLAB software provides a straightforward way to output data to the Command Window. Finally, saving data is an important task for their use in future works. Therefore, in MATLAB environment we can save data and preserve their structures and variables.

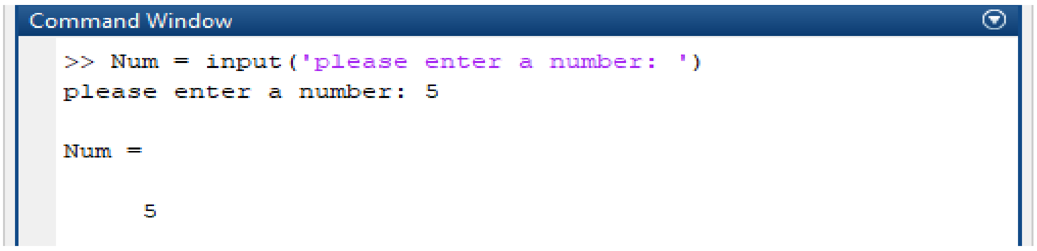
In this practical work, our objective is to investigate different functions that enable the effective management of the outlined operations (reading, displaying, and saving data). We will not only understand the mechanics of these functions but also reinforce our understanding through simple and comprehensive examples.

1. Reading data in MATLAB

In MATLAB, it is possible to read the different data structures from several sources, including files and user input. The next subtitles present the most used function to read data.

1.1 The function `Input()`:

This function allows reading data structure entered by users**.** The syntax of this function is expressed as follows:

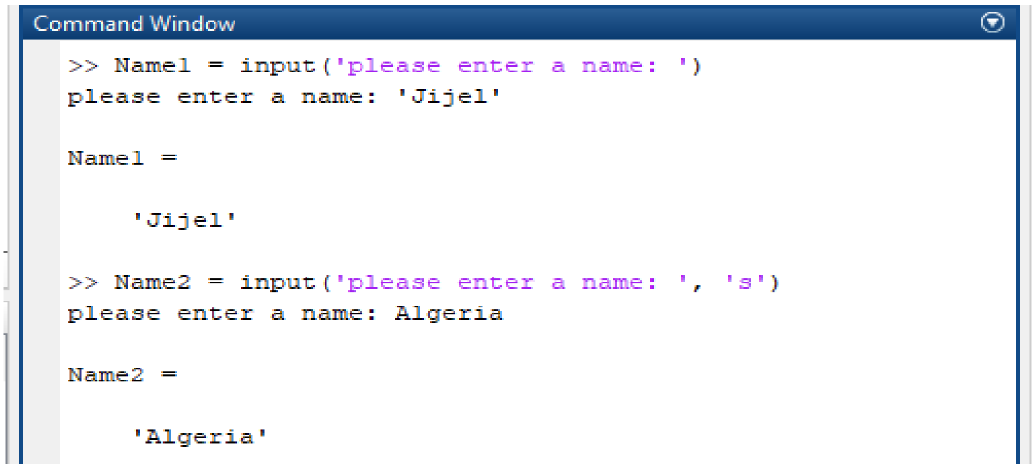


```
>> Num = input('please enter a number: ')
please enter a number: 5

Num =

     5
```

In this example, the user should enter a variable, and the variable is stored in the variable '`Num`'. If the user needs to enter a text instead of a number, it is necessary to use quotation marks around the string. In a more appropriate way, we must add `'s'` argument in the use of the input function. The two possible ways are indicated below:



Command Window

```
>> Name1 = input('please enter a name: ')
please enter a name: 'Jijel'

Name1 =

    'Jijel'

>> Name2 = input('please enter a name: ', 's')
please enter a name: Algeria

Name2 =

    'Algeria'
```

1.2 The function `Load ()`:

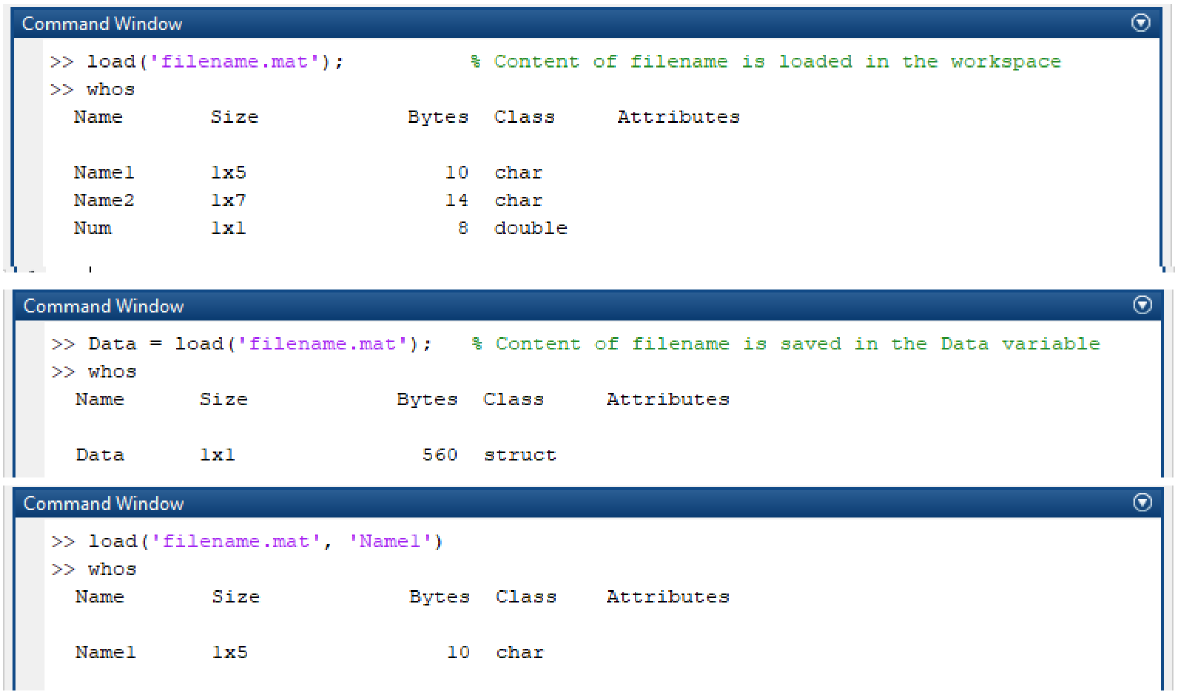
This function is used to read data structure from a file, which can be a MATLAB file (.mat file), text file, Excel file or others and store it in the workspace or in a defined user variable. The basic syntax of this function is indicated below:

`load('file-name')`: Read data from the indicated file `'file-name'`.

`Data =load('file-name')`: Read data and save it in variable `Data`.

`load('file-name', 'variables')`: Read specified variables from a .mat file.

Figure.1 displays the use of the load function.



Command Window

```
>> load('filename.mat');          % Content of filename is loaded in the workspace
>> whos
  Name        Size             Bytes  Class     Attributes

  Name1       1x5                 10  char
  Name2       1x7                 14  char
  Num         1x1                  8  double
```

Command Window

```
>> Data = load('filename.mat');   % Content of filename is saved in the Data variable
>> whos
  Name        Size             Bytes  Class     Attributes

  Data        1x1                560  struct
```

Command Window

```
>> load('filename.mat', 'Name1')
>> whos
  Name        Size             Bytes  Class     Attributes

  Name1       1x5                 10  char
```

Figure.1 Use of the `load()` function

In the first case, the `load()` function reads the data structure located in `'filename.mat'` and stores it into the workspace. However, the acquired data are saved into `Data` variable in the second case. The last example presents the use of `load()` function to load a specific variable from a .mat file.

In addition, we can read the data located in external file using the MATLAB menu as indicated in Figure.2.

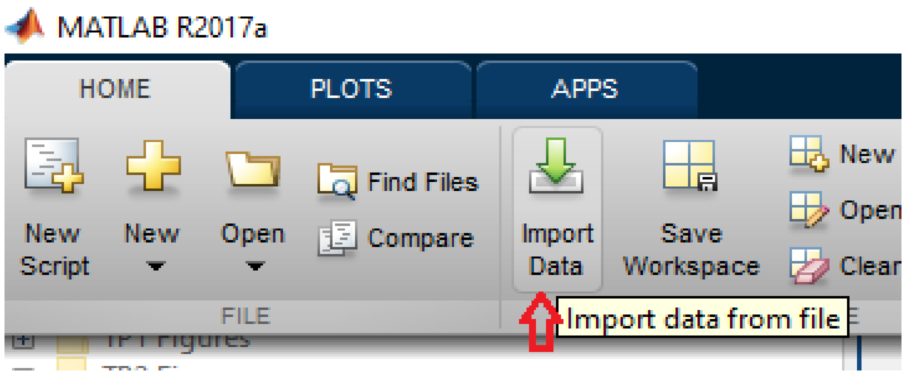


Figure 2: Loading data using MATLAB menu

2. Saving data in MATLAB

In MATLAB, saving data presents a crucial operation of maintaining the content of different data structures to a file for future use. Saving data in MATLAB helps to preserve the results of programs and share them with others or follow computations in subsequent analysis.

There are various methods or manners used to save data structure in MATLAB regarding saving workspace variable into MAT file, saving data to text or spreadsheet files. Here are some common ways to save data in MATLAB:

2.1 Saving data in binary file format

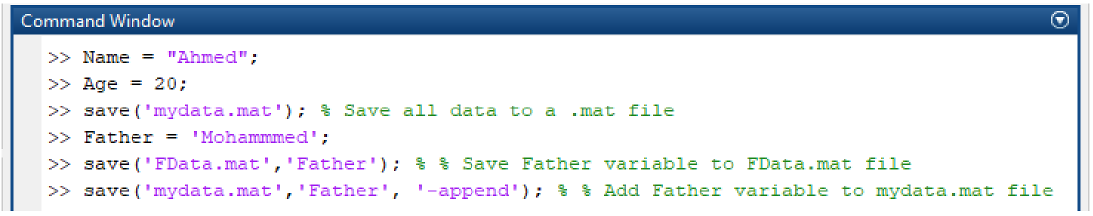
To save the variables of the workspace in binary format (.mat file), we use the following functions:

`save ('file-name.mat') or save file-name.mat` : saves all variables located in the workplace;

`save ('file-name', 'variable1', 'variable2') or save file-name, variable1, variable2`: saves only the workspace variables you list after the `filename: 'variable1'` and `'variable2'`.

`load('file-name','variables','-append')` adds variables to the existing file-name. If a variable already exists in the selected file, the save function overwrites it with the new value in the workspace.

The following command window shows an example of using the `save()` command.



```
>> Name = "Ahmed";
>> Age = 20;
>> save('mydata.mat'); % Save all data to a .mat file
>> Father = 'Mohammmed';
>> save('FData.mat','Father'); % % Save Father variable to FData.mat file
>> save('mydata.mat','Father', '-append'); % % Add Father variable to mydata.mat file
```

In addition, we can save the data into file by using the menu bar of MATLAB or contextual menu in the workspace part, as presented in the figure 2:

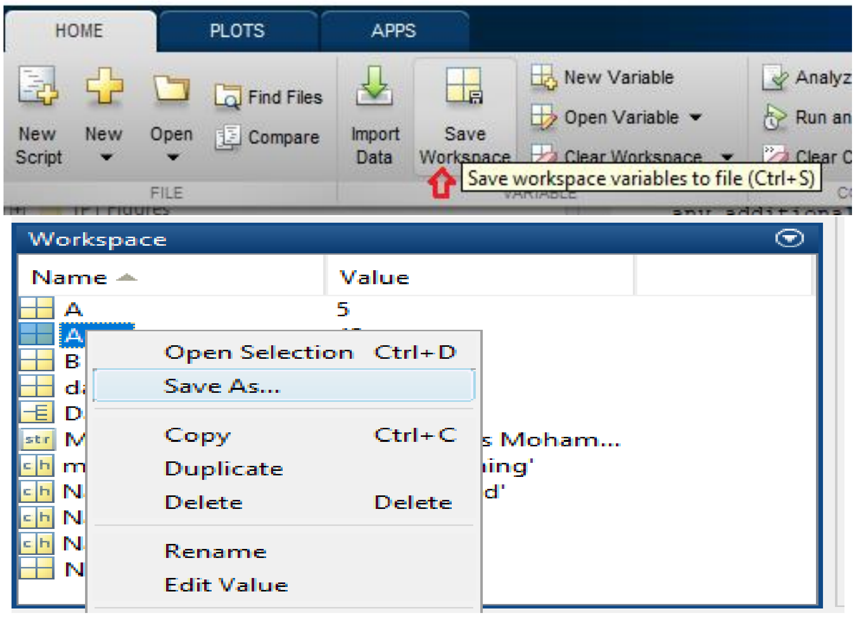


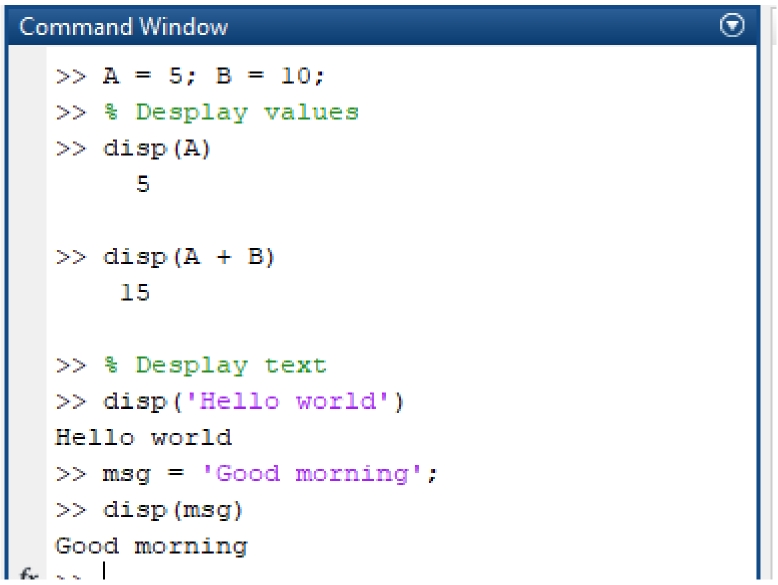
Figure 2:Save data using MATLAB menu

3. Displaying data in MATLAB

Displaying data in MATLAB is an operation generally performed through using functions to show information or results to the user. MATLAB offers various methods for displaying data. The next points present the common ways to display data in MATLAB.

3.1 The function `disp( )`

The `disp()` function is employed in MATLAB to display different kinds of data structure, such as numbers and text. However, `disp()` does not allow formatting. The next command window shows the use of `disp()` function.



```
>> A = 5; B = 10;
>> % Desplay values
>> disp(A)
     5

>> disp(A + B)
    15

>> % Desplay text
>> disp('Hello world')
Hello world
>> msg = 'Good morning';
>> disp(msg)
Good morning
>> |
```

3.2 The function `fprintf ()`

The `fprintf()` function is used for displaying different data structures, where the main characteristic of this function is controlling and determining the format and layout of the displayed outputs or results. The basic syntax of `fprinf()` is as follows:

`fprintf(formatSpec,A1,...,An)`

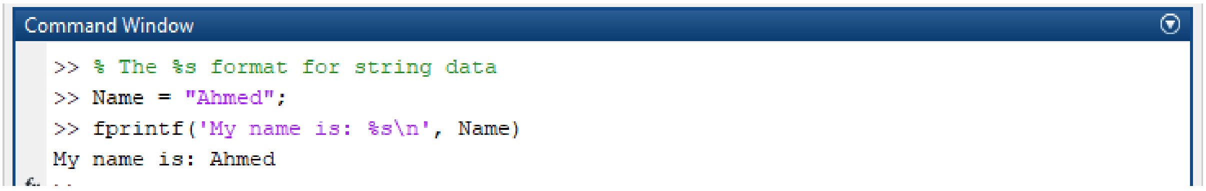
where:

`formatSpec` is a set of characters that determine the format of the output and can include combinations of `%` character and a conversion character such as `d, i, o, u, x, f, e, g, c, or s.`

`A1,...,An` presents the variables or the data structure needed to include the formatted output.

The next points clarify the most used formatting options:

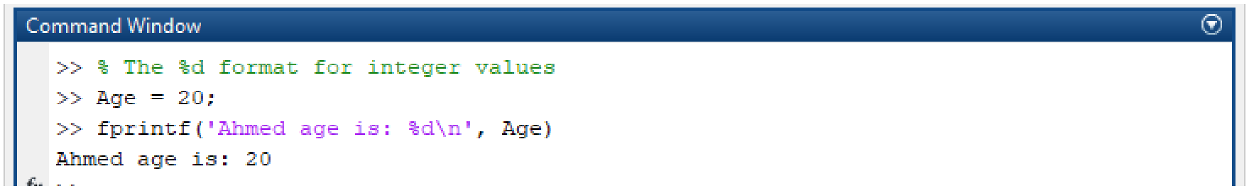
a) The `%s` format: used to format and display string data, such as words, sentences, or text messages. If we display only one character, we can use **`%c`** format. A practical example is showcased in what follows:



```
>> % The %s format for string data
>> Name = "Ahmed";
>> fprintf('My name is: %s\n', Name)
My name is: Ahmed
```

In this example, we use `%s` to insert the value of the `Name` variable into the formatted output. The `\n` argument is used to move down to the next line.

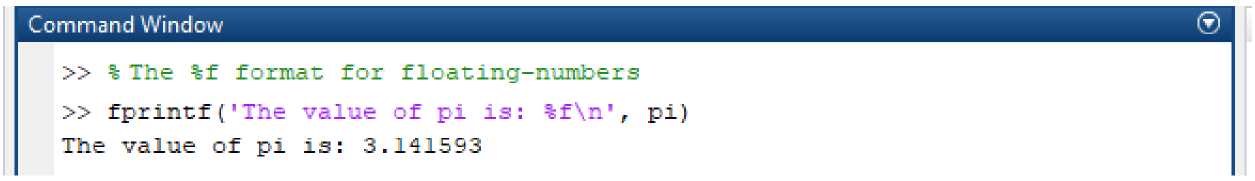
b) **The %d format**: used to format and display integer values in a specified format as indicated in the next example.



```
>> % The %d format for integer values
>> Age = 20;
>> fprintf('Ahmed age is: %d\n', Age)
Ahmed age is: 20
```

In this example, we use `%d` to format and display the variable `Age`. The result of this program displays a message such as "`Ahmed age is 20`", where `%d` is replaced by the value of the `Age` variable.

c) **The %f format**: This format is used to insert, control and display the format of floating-numbers. In other manner, the `%f` format helps to display the real numbers with decimal points in the results. The use of this format is indicated below:

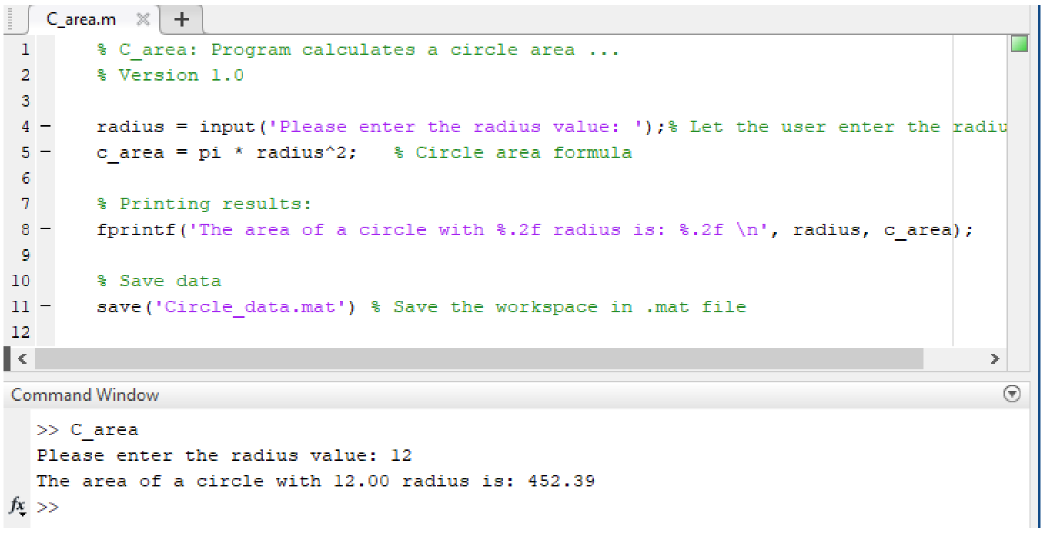


```
>> % The %f format for floating-numbers
>> fprintf('The value of pi is: %f\n', pi)
The value of pi is: 3.141593
```

In this example, we use the `%f` to determine the format of floating-numbers. The results of this program show a message such as "`The value of pi is:  3.141593`" where `%f` is replaced by the value of the `pi` constant.

4. Script with input, output and save functions

In our first script, we have created a program that calculates the area of a circle. Now, we improve our program by using the input, output and save functions to let the user enter the value of the radius, display the calculated area in the screen and save the workspace into a `.mat` file. Figure.3 shows such modifications and the execution of the script.



```
% C_area: Program calculates a circle area ...
% Version 1.0

radius = input('Please enter the radius value: ');% Let the user enter the radiu
c_area = pi * radius^2;    % Circle area formula

% Printing results:
fprintf('The area of a circle with %.2f radius is: %.2f \n', radius, c_area);

% Save data
save('Circle_data.mat') % Save the workspace in .mat file
```

```
>> C_area
Please enter the radius value: 12
The area of a circle with 12.00 radius is: 452.39
>>
```

Figure. 3. Circle area program with input and output functions

5. Practice

1. Open MATLAB and create a new script file.

Write the program that computes the following formula (using the input and the output functions):

$$R1 = 4\pi/2 \times a$$

$$R2 = cosinus\ (6\pi/10) \times b$$

- Save all variables located in workspace in file called program
- Save only the final results in file called result.

2. Based on the previous program :
   - Read the saved files ;
   - Display the entered variables ;
   - Display the obtained results.

3. Write a script that will calculate the surface of a cylinder given by:

$Cylinder\ surface = 2\pi r^2 + 2\pi rh$ using the input and the output functions.

where r and h are the radius and the height of the cylinder respectively.

- Using the developed script calculate the surface of the indicated cylinders below:
  - Cylinder1 : $r = 5.53,\ h = 3.8$;
  - Cylinder2 : $r = 8.2,\ h = 10.45$;
  - Cylinder3 : $r = 120,\ h = 38.22$.
- Save the obtained results in a .mat file ;
- Clear all data;
- Using a variable, read from the saved data only the heights of the different cylinders.