MATLAB - DATA IMPORT

Importing data in MATLAB means loading data from an external file. The **importdata** function allows loading various data files of different formats. It has the following five forms −

S.N.	Function and Description
1	**A = importdata*filename*** Loads data into array A from the file denoted by *filename*.
2	**A = importdata*' −pastespecial'*** Loads data from the system clipboard rather than from a file.
3	**A = importdata___, delimiterIn** Interprets *delimiterIn* as the column separator in ASCII file, filename, or the clipboard data. You can use *delimiterIn* with any of the input arguments in the above syntaxes.
4	**A = importdata___, delimiterIn, headerlinesIn** Loads data from ASCII file, filename, or the clipboard, reading numeric data starting from line *headerlinesIn+1*.
5	**[A, delimiterOut, headerlinesOut] = importdata___** Returns the detected delimiter character for the input ASCII file in *delimiterOut* and the detected number of header lines in *headerlinesOut*, using any of the input arguments in the previous syntaxes.

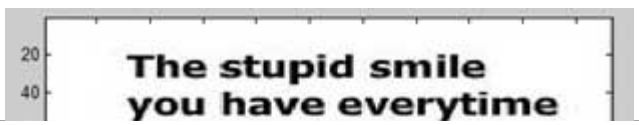
> *By default, Octave does not have support for importdata function, so you will have to search and install this package to make following examples work with your Octave installation.*

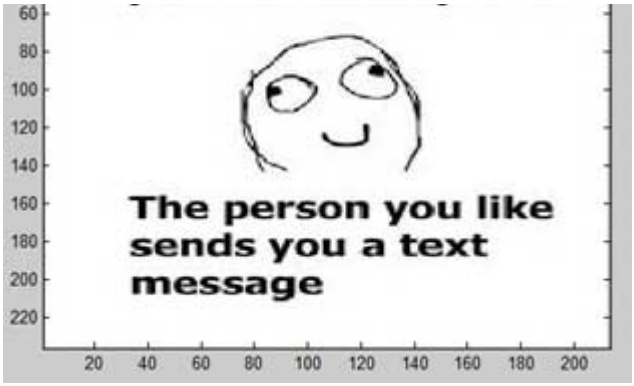
Example 1

Let us load and display an image file. Create a script file and type the following code in it:

```
filename = 'smile.jpg';
A = importdata(filename);
image(A);
```

When you run the file, MATLAB displays the image file. However, you must store it in the current directory.





Example 2

In this example, we import a text file and specify Delimiter and Column Header. Let us create a space-delimited ASCII file with column headers, named *weeklydata.txt*.

Our text file weeklydata.txt looks like this −

```
SunDay   MonDay   TuesDay   WednesDay   ThursDay   FriDay   SaturDay
95.01    76.21    61.54      40.57       55.79     70.28     81.53
73.11    45.65    79.19      93.55       75.29     69.87     74.68
60.68    41.85    92.18      91.69       81.32     90.38     74.51
48.60    82.14    73.82      41.03       0.99      67.22    93.18
89.13    44.47    57.63      89.36       13.89     19.88     46.60
```

Create a script file and type the following code in it −

```
filename = 'weeklydata.txt';
delimiterIn = ' ';
headerlinesIn = 1;
A = importdata(filename,delimiterIn,headerlinesIn);
% View data
for k = [1:7]
   disp(A.colheaders{1, k})
   disp(A.data(:, k))
   disp(' ')
end
```

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

```
SunDay
   95.0100
   73.1100
   60.6800
   48.6000
   89.1300

MonDay
   76.2100
   45.6500
   41.8500
   82.1400
   44.4700

TuesDay
   61.5400
   79.1900
   92.1800
   73.8200
   57.6300

WednesDay
   40.5700
   93.5500
   91.6900
```

```
   41.0300
   89.3600

ThursDay
   55.7900
   75.2900
   81.3200
    0.9900
   13.8900

FriDay
   70.2800
   69.8700
   90.3800
   67.2200
   19.8800

SaturDay
   81.5300
   74.6800
   74.5100
   93.1800
   46.6000
```

Example 3

In this example, let us import data from clipboard.

Copy the following lines to the clipboard −

Mathematics is simple

Create a script file and type the following code −

```
A = importdata('-pastespecial')
```

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

```
A =
    'Mathematics is simple'
```

Low-Level File I/O

The *importdata* function is a high-level function. The low-level file I/O functions in MATLAB allow the most control over reading or writing data to a file. However, these functions need more detailed information about your file to work efficiently.

MATLAB provides the following functions for read and write operations at the byte or character level −

Function	Description
fclose	Close one or all open files
feof	Test for end-of-file
ferror	Information about file I/O errors
fgetl	Read line from file, removing newline characters
fgets	Read line from file, keeping newline characters
fopen	Open file, or obtain information about open files
fprintf	Write data to text file

fread	Read data from binary file
frewind	Move file position indicator to beginning of open file
fscanf	Read data from text file
fseek	Move to specified position in file
ftell	Position in open file
fwrite	Write data to binary file

Import Text Data Files with Low-Level I/O

MATLAB provides the following functions for low-level import of text data files −

- The **fscanf** function reads formatted data in a text or ASCII file.
- The **fgetl** and **fgets** functions read one line of a file at a time, where a newline character separates each line.
- The **fread** function reads a stream of data at the byte or bit level.

Example

We have a text data file 'myfile.txt' saved in our working directory. The file stores rainfall data for three months; June, July and August for the year 2012.

The data in myfile.txt contains repeated sets of time, month and rainfall measurements at five places. The header data stores the number of months M; so we have M sets of measurements.

The file looks like this −

```
Rainfall Data
Months: June, July, August

M=3
12:00:00
June-2012
17.21  28.52  39.78  16.55 23.67
19.15  0.35   17.57  NaN   12.01
17.92  28.49  17.40  17.06 11.09
9.59   9.33   NaN    0.31  0.23
10.46  13.17  NaN    14.89 19.33
20.97  19.50  17.65  14.45 14.00
18.23  10.34  17.95  16.46 19.34
09:10:02
July-2012
12.76  16.94  14.38  11.86 16.89
20.46  23.17  NaN    24.89 19.33
30.97  49.50  47.65  24.45 34.00
18.23  30.34  27.95  16.46 19.34
30.46  33.17  NaN    34.89  29.33
30.97  49.50  47.65  24.45 34.00
28.67  30.34  27.95  36.46 29.34
15:03:40
August-2012
17.09  16.55  19.59  17.25 19.22
17.54  11.45  13.48  22.55 24.01
NaN    21.19  25.85  25.05 27.21
26.79  24.98  12.23  16.99 18.67
17.54  11.45  13.48  22.55 24.01
NaN    21.19  25.85  25.05 27.21
26.79  24.98  12.23  16.99 18.67
```

We will import data from this file and display this data. Take the following steps −

- Open the file with **fopen** function and get the file identifier.
- Describe the data in the file with **format specifiers**, such as '**%s**' for a string, '**%d**' for an integer, or '**%f**' for a floating-point number.
- To skip literal characters in the file, include them in the format description. To skip a data field, use an asterisk ′ ∗′ in the specifier.

  For example, to read the headers and return the single value for M, we write:

```
M = fscanf(fid, '%*s %*s\n%*s %*s %*s %*s\nM=%d\n\n', 1);
```

- By default, **fscanf** reads data according to our format description until it does not find any match for the data, or it reaches the end of the file. Here we will use for loop for reading 3 sets of data and each time, it will read 7 rows and 5 columns.
- We will create a structure named *mydata* in the workspace to store data read from the file. This structure has three fields - *time*, *month*, and *raindata* array.

Create a script file and type the following code in it –

```
filename = '/data/myfile.txt';
rows = 7;
cols = 5;

% open the file
fid = fopen(filename);

% read the file headers, find M (number of months)
M = fscanf(fid, '%*s %*s\n%*s %*s %*s %*s\nM=%d\n\n', 1);

% read each set of measurements
for n = 1:M
   mydata(n).time = fscanf(fid, '%s', 1);
   mydata(n).month = fscanf(fid, '%s', 1);

   % fscanf fills the array in column order,
   % so transpose the results
   mydata(n).raindata  = ...
      fscanf(fid, '%f', [rows, cols]);
end
for n = 1:M
   disp(mydata(n).time), disp(mydata(n).month)
   disp(mydata(n).raindata)
end

% close the file
fclose(fid);
```

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

```
12:00:00
June-2012
   17.2100   17.5700   11.0900   13.1700   14.4500
   28.5200       NaN    9.5900       NaN   14.0000
   39.7800   12.0100    9.3300   14.8900   18.2300
   16.5500   17.9200       NaN   19.3300   10.3400
   23.6700   28.4900    0.3100   20.9700   17.9500
   19.1500   17.4000    0.2300   19.5000   16.4600
    0.3500   17.0600   10.4600   17.6500   19.3400

09:10:02
July-2012
   12.7600       NaN   34.0000   33.1700   24.4500
   16.9400   24.8900   18.2300       NaN   34.0000
   14.3800   19.3300   30.3400   34.8900   28.6700
   11.8600   30.9700   27.9500   29.3300   30.3400
```

```
   16.8900   49.5000   16.4600   30.9700   27.9500
   20.4600   47.6500   19.3400   49.5000   36.4600
   23.1700   24.4500   30.4600   47.6500   29.3400

15:03:40
August-2012
   17.0900   13.4800   27.2100   11.4500   25.0500
   16.5500   22.5500   26.7900   13.4800   27.2100
   19.5900   24.0100   24.9800   22.5500   26.7900
   17.2500       NaN   12.2300   24.0100   24.9800
   19.2200   21.1900   16.9900       NaN   12.2300
   17.5400   25.8500   18.6700   21.1900   16.9900
   11.4500   25.0500   17.5400   25.8500   18.6700
```