load

Load workspace variables from disk

Syntax

```
load
load filename
load filename X Y Z ...
load filename -regexp expr1 expr2 ...
load -ascii filename
load -mat filename
S = load('arg1', 'arg2', 'arg3', ...)
```

Description

`load` loads all the variables from the MAT−file `matlab.mat`, if it exists, or returns an error if the file doesn't exist.

`load filename` loads all the variables from the file specified by `filename`. `filename` is an unquoted string specifying a file name, and can also include a file extension and a full or partial pathname. If `filename` has no extension, `load` looks for a file named `filename.mat` and treats it as a binary MAT−file. If `filename` has an extension other than `.mat`, `load` treats the file as ASCII data.

`load filename X Y Z ...` loads just the specified variables `X`, `Y`, `Z`, etc. from the MAT−file. The wildcard `'*'` loads variables that match a pattern (MAT−file only).

`load filename -regexp expr1 expr2 ...` loads those variables that match any of the regular expressions given by `expr1`, `expr1`, etc.

`load -ascii filename)` forces `load` to treat the file as an ASCII file, regardless of file extension. If the file is not numeric text, `load` returns an error.

`load -mat filename)` forces `load` to treat the file as a MAT−file, regardless of file extension. If the file is not a MAT−file, `load` returns an error.

`S = load('arg1', 'arg2', 'arg3', ...)` calls `load` using MATLAB *function syntax*, (as opposed to the MATLAB *command syntax* that has been shown thus far). You can use function syntax with any form of the `load` command shown above, replacing `arg1`, `arg2`, etc. with the arguments shown. For example,

```
S = load('myfile.mat', '−regexp', '^Mon', '^Tue')
```

To specify a command line option, such as `−mat`, with the functional form, specify the option as a string argument, and include the hyphen. For example,

```
load('myfile.dat', '−mat')
```

Function syntax enables you to assign values returned by `load` to an output variable. You can also use function syntax when loading from a file having a name that contains space characters, or a filename that is stored in a variable.

If the file you are loading from is a MAT−file, then output `S` is a structure containing fields that match the variables retrieved. If the file contains ASCII data, then `S` is a double−precision array.

Remarks

For information on any of the following topics related to saving to MAT−files, see Importing Data from MAT−Files in the "MATLAB Programming" documentation:

- Previewing MAT−file contents
- Loading binary data
- Loading ASCII data

You can also use the Current Directory browser to view the contents of a MAT−file without loading it−−see Viewing Information About M−Files and MAT−Files.

Examples

Example 1. Loading From a Binary MAT−file

To see what is in the MAT−file prior to loading it, use `whos -file`:

```
whos -file mydata.mat
  Name           Size             Bytes  Class

  javArray      10x1                     java.lang.Double[][
  spArray        5x5                84  double array (spars
  strArray       2x5               678  cell array
  x              3x2x2              96  double array
  y              4x5              1230  cell array
```

Clear the workspace and load it from MAT−file `mydata.mat`:

```
clear
load mydata

whos
  Name           Size             Bytes  Class

  javArray      10x1                     java.lang.Double[][
  spArray        5x5                84  double array (spars
  strArray       2x5               678  cell array
  x              3x2x2              96  double array
  y              4x5              1230  cell array
```

Example 2. Loading a List of Variables

You can use a comma−separated list to pass the names of those variables you want to load from a file. This example generates a comma−separated list from a cell array:

```
filename = 'myfile.mat';
whos -file filename
```

```
  Name          Size                    Bytes  Class

  AName         1x24                       48  char array
  AVal          1x1                         8  double array
  BName         1x24                       48  char array
  BVal          1x1                         8  double array
  CVal          5x5                        84  double array (spars
  DArr          2x5                       678  cell array
```

```
filevariables = {'AName', 'BVal', 'DArr'};
load(filename, filevariables{:});
```

The second part of this example generates a comma−separated list from the name field of a structure array, and loads the first ten variables from the specified file:

```
filename = 'myfile.mat';
vars = whos('−file', filename);
load(filename, vars(1:10).name);
```

Example 3. Loading From an ASCII File

Create several 4−column matrices and save them to an ASCII file:

```
a = magic(4);  b = ones(2, 4) * −5.7;  c = [8 6 4 2];
save −ascii mydata.dat
```

Clear the workspace and load it from the file `mydata.dat`. If the filename has an extension other than `.mat`, MATLAB assumes that it is ASCII:

```
clear
load mydata.dat
```

MATLAB loads all data from the ASCII file, merges it into a single matrix, and assigns the matrix to a variable named after the filename:

```
mydata
mydata =
   16.0000    2.0000    3.0000   13.0000
    5.0000   11.0000   10.0000    8.0000
    9.0000    7.0000    6.0000   12.0000
    4.0000   14.0000   15.0000    1.0000
   −5.7000   −5.7000   −5.7000   −5.7000
   −5.7000   −5.7000   −5.7000   −5.7000
    8.0000    6.0000    4.0000    2.0000
```

Example 4. Using Regular Expressions

Using regular expressions, load from MAT−file `mydata.mat` those variables with names that begin with `Mon`, `Tue`, or `Wed`:

```
load('mydata', '−regexp', '^Mon|^Tue|^Wed');
```

Here is another way of doing the same thing. In this case, there are three separate expression arguments:

```
load('mydata', '−regexp', '^Mon', '^Tue', '^Wed');
```

See Also

clear, fprintf, fscanf, partialpath, save, spconvert, who

listdlg load (COM)