# Creating MATLAB mfiles  EAS 199B
# Review of screencast

## Introduction

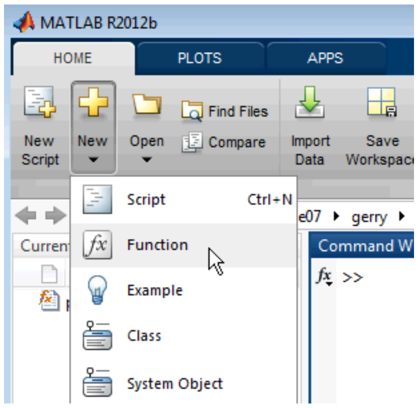
MATLAB mfiles are plain text files that contain MATLAB commands. A script mfile is just a list of commands. Running a script is like typing the contents of the script mfile into the command window. A function mfile is a list of commands with an defined list of input and output arguments. The commands executed in a function mfile are isolated from the command window. Only the input and output arguments of the function are communicated with the command window.

Function mfiles are recommended over script mfiles because the input and output arguments make the function reusable in different contexts. In addition, the user does not need to worry about local variables in the function corrupting the variables in the command window. Function mfiles promote good coding practices of reuse and isolation of local variables.

As a simple example, we will create a script to plot $y = \sin(x)$. There is no need to create a function for this task. The commands can be entered directly into the command window. However, we begin with a simple and familiar set of commands so we can focus on the mechanics of creating and using a function, and not worry so much what that function actually does.

## 1. Creating an mfile

Open the editor by clicking on the `New` button in the upper left corner of the tab bar, and select `Function` from the pop-up list.



This opens up a new window called the editor.

## 2. A simple script to plot y = sin(x)

For this simple example, the function will have no input or output arguments, so replace the first line in the function with

```
function myplot
```

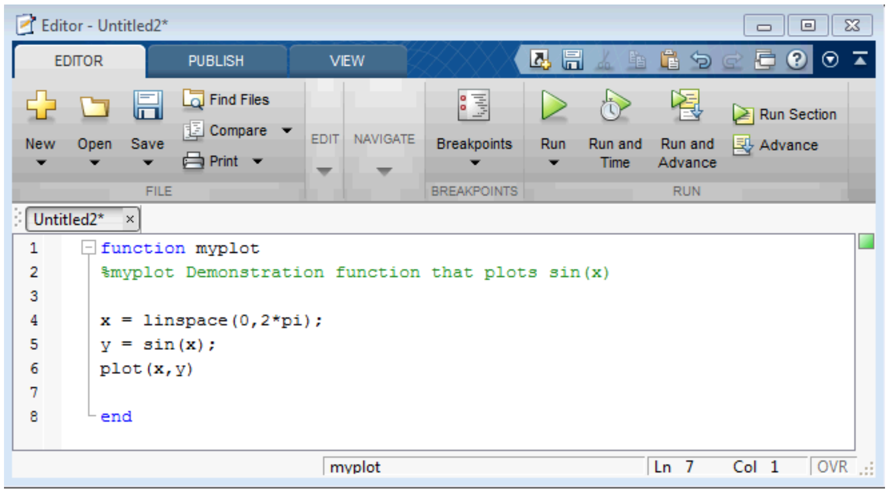
Change the second and third lines to

```
%myplot  Demonstration function that plots sin(x)
```

and add the following commands

```
x = linspace(0,2*pi);
y = sin(x)
plot(x,y)
```

The completed function looks like this:



```
function myplot
%myplot Demonstration function that plots sin(x)

x = linspace(0,2*pi);
y = sin(x);
plot(x,y)

end
```

## 3. Saving the mfile

Click the Save button in the toolbar and select Save As …

Navigate to your desired folder.

Notice that the file name ends with ".m"

## 4. Run the mfile

After you have the file, type its name at the command prompt

```
myplot
```

You may get this error message

```
Undefined function or variable 'myplot'.
```

If so, you will need to change the *current working directory* to the directory (a.k.a. folder) to the directory that contains your mfile. Refer to the *Further Reading* section at the end of these notes for links to documentation on the Mathworks web site.

## 5. Check your understanding: Restart MATLAB

Quit MATLAB

Restart MATLAB

Run myplot

If necessary, navigate to the directory that contains myplot

## 6. Add an input argument

Enable the user (via the command window) to change the end of the range for the x values. Add "`(xmax)`" to end of the first line of the `myplot.m`.

```
function myplot(xmax)

x = linspace(0,xmax);
y = sin(x);
plot(x,y)

end
```

Run the modified function from the command line

```
myplot(5*pi);
```

where `5*pi` could be replaced by any value

## 7. Add a second input argument

Change the end of the first line of `myplot.m` from "`(xmax)`" to "`(xmax,npoints)`" and add npoints as the third argument to the `linspace` command.

```
function myplot(xmax,npoints)

x = linspace(0,xmax,npoints);
y = sin(x);
plot(x,y)

end
```

Run the modified function from the command line

```
myplot(5*pi,20);
```

where `5*pi` and `20` can be replaced by other values.

## 8. Add a return argument

In the first line of `myplot.m` insert "`ymax =`" before "myplot". In the body of the function, after the values of `y` are assigned, add the line `ymax = max(y);`

```
function ymax = myplot(xmax,npoints)

x = linspace(0,xmax,npoints);
y = sin(x);
ymax = max(y);
plot(x,y)

end
```

Run it from the command line

```
ymax = myplot(5*pi,20);
```

You can use any variable name you want to receive the returned value

```
maximum_y = myplot(5*pi,20);
```

## 9. Add a second return argument

```
function [ymax,yave] = myplot(xmax,npoints)

x = linspace(0,xmax,npoints);
y = sin(x);
ymax = max(y);
yave = mean(y);
plot(x,y)

end
```

Run it from the command line

```
[y_max,y_ave] = myplot(5*pi,20)
```

Note that `y_max` and `y_ave` are not the same names (`ymax` and `yave`) that are used in the function definition line (the first line) of `myplot.m`. All input and output arguments are positional, i.e., when values are communicated between the function and the command window, only the order of the arguments, not the names, matter. This is a very important principle since it allows a function to be used in different situations and the programmer who uses the function does not need to know about the internal variable names of the function.

## Further reading

Making files and folders accessible to MATLAB:

http://www.mathworks.com/help/matlab/matlab_env/making-files-and-folders-accessible-to-matlab.html

Understanding file locations in MATLAB:

http://www.mathworks.com/help/matlab/matlab_env/understanding-file-locations-in-matlab.html