Ex. No : 9 Automating Commands with Scripts, Using the Command History, Creating script files, Running scripts, Dividing code into sections and Publishing scripts

Automating Commands with Scripts

A script file is an m-file that contains a sequence of instructions but is not a function. Unlike a function, a script file shares its workspace with the current directory. A script file is created by writing lines of code just as you would at the command prompt. A script file is run when you press the run button  in the top toolbar. When the script file is run, MATLAB executes the instructions in order as if you typed them into the command prompt one at a time.

Example:

Write a script file that computes properties of a cylinder for a given radius and height.

```
clc
clear all
close all

% declare cylinder specs
r = 5;
h = 10;

% compute the volume of cylinder
V = pi*r^2*h

% compute lateral surface area of cylinder
s = 2*pi*r*h

% compute total surface area of cylinder
S = s + 2*pi*r^2
```

If you had a new cylinder, you could change r and h and rerun the script file to compute the new properties.

WARNING!

Script files share their workspace with the current directory, so be careful that variables assigned in the script file are not overwriting other variables with the same name.

TIP!

Always have a clc, clear all, and close all at the beginning of every script file. This is because the script shares the workspace with the current directory, and it is good practice to make sure that your script file is not using variables from old MATLAB sessions.

It is natural to wonder when a script file is more appropriate than a function or vice versa. Functions are most useful when a task needs to be done many times. Script files are useful when you need to perform a sequence of instructions only a few times in a highly context-specific situation. Some examples might be producing a complicated plot or trying something new to see if it is worth writing a function for it.

You can organize code in script files into cells (not to be confused with cell arrays), which can be run without running the rest of the script. This is not to be confused with cell arrays. A cell is a piece of script code that can be run individually. Code cells can be created by double commenting, or using two %%. You can execute a cell by clicking the evaluate-cell button  or the evaluate-cell-and-advance button  .

TRY IT!

```
% Script File for Computing Properties of a Cylinder
clc
clear all
close all

% declare cylinder specs
r = 5;
h = 10;

% compute the volume of cylinder
V = pi*r^2*h

% compute lateral surface area of cylinder
s = 2*pi*r*h

% compute total surface area of cylinder
S = s + 2*pi*r^2
```

Organize the previous script file into cells. Execute the cells one at a time using cell mode.

You can turn cell mode off or on under Cell → Disable Cell Mode.

Creating script files

Scripts are the simplest kind of code file because they have no input or output arguments. They are useful for automating series of MATLAB commands, such as computations that you have to perform repeatedly from the command line or series of commands you have to reference.

You can create a new script in the following ways:

- Highlight commands from the Command History, right-click, and select **Create Script**.
- On the **Home** tab, click the **New Script**  button.
- Use the edit function. For example, edit *new_file_name* creates (if the file does not exist) and opens the file *new_file_name*. If *new_file_name* is unspecified, MATLAB opens a new file called Untitled.

After you create a script, you can add code to the script and save it. For example, you can save this code that generates random numbers from 0 through 100 as a script called numGenerator.m.

Example:

```
columns = 10000;
rows = 1;
bins = columns/100;

rng(now);
list = 100*rand(rows,columns);
histogram(list,bins)
```

Save your script and run the code using either of these methods:

- Type the script name on the command line and press **Enter**. For example, to run the numGenerator.m script, type numGenerator.

- On the **Editor** tab, click the **Run**  button.

You also can run the code from a second code file. To do this, add a line of code with the script name to the second code file. For example, to run the numGenerator.m script from a second code file, add the line numGenerator; to the file. MATLAB runs the code in numGenerator.m when you run the second file.

When execution of the script completes, the variables remain in the MATLAB workspace. In the numGenerator.m example, the variables columns, rows, bins, and list remain in the workspace. To see a list of variables, type whos at the command prompt. Scripts share the base workspace with your interactive MATLAB session and with other scripts.

Running scripts

Syntax

```
run(scriptname)
```

Description

run(scriptname) runs the MATLAB script specified by scriptname.

Example

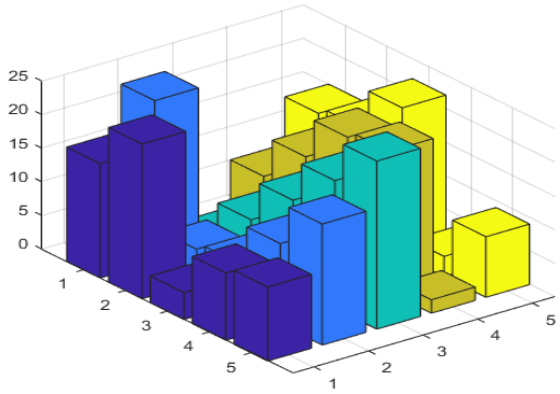
```
tmp = tempname;
mkdir(tmp)
```

Write MATLAB code to a file in the folder.

```
newFile = fullfile(tmp,'ANewFile.m');
fid = fopen(newFile,'w');
fprintf(fid,'Z = magic(5);\n');
fprintf(fid,'b = bar3(Z);\n');
fclose(fid);
```

Run the script.

run(newFile)



Input Arguments

scriptname — Full or relative script path (character vector | string scalar)

Full or relative script path to a MATLAB script, specified as a character vector or string scalar. scriptname can specify any file type that MATLAB can execute, such as MATLAB script files, Simulink models, or MEX-files.

Example: scriptname = 'myScript'

Example: scriptname = 'anotherScript.m'

Example: scriptname = 'oneMoreScript.mlx'

Tips:

- run can execute a script not on the MATLAB path if its input argument specifies the path to the script. To run a script by simply entering its name, you should use cd to navigate to the appropriate folder or addpath to add the folder to the MATLAB search path.
- scriptname can access any variables in the current workspace.
- run changes to the folder that contains the script, executes it, and resets back to the original folder. If the script itself changes folders, then run does not revert to the original folder, unless scriptname changes to the folder in which this script resides.
- If scriptname corresponds to both a .m file and a P-file residing in the same folder, then run executes the P-file. This occurs even if you specify scriptname with a .m extension.

- If a script is not on the MATLAB path, executing the run command caches the script. In the same session and after calling run, you can edit the script using an external editor. Call clear scriptname before calling run again to use the changed version of the script rather than the cached version. If you edit the script with the MATLAB editor, run executes the changed version and there is no need to call clear scriptname.

Work to be done:

1. Create a simple script which adds the squares of both x and y. In your terminal set x = 2 and y = 4. What happens if you do or don't include "clear all" in your MATLAB script? (Note : Do a help on "clear")
2. In a new MATLAB script named zero test.m make a random matrix which has dimension 1000 × 1000. Create a matrix of identical size of zeros. Add, subtract, and multiply these matrices together. In full sentences in comments, at the top of your script, describe each "built-in" MATLAB function used and the intent of this script. In the first three lines of these comments place your name, the script name, and the date (Note : Do a help on "zeros", and a help on "rand".)
3. Write a script that, given in input a natural number k, compute the first k elements of the Fibonacci series, given by the recurrence formula:

$$F_0 = 1, F_1 = 1, \quad F_i = F_{i-1} + F_{i-2} \quad \forall i \geq 2$$

4. Create a vector in your script with a list of dates:
   1. clear
   2. dates = [1015 1066 1660 1814 1905 2014]
5. Create the vectors v1=(1,2,3,4,5) and v2=(10,20,30,40,50) and:

1. sum v1 and v2 and store the result in vs
2. subtract v2 from v1 and store the result in vd
3. compute the scalar product of v1 and v2 and store the result in s.