Functions

Jiseok Chae

Department of Mathematical Sciences
KAIST

Week 3

Contents

The Command Window works well, when you want to execute commands one by one.

However, to do more complex tasks, Command Window is usually not the best choice.

Better way is to write a *program*, i.e. a file with a collection of commands.

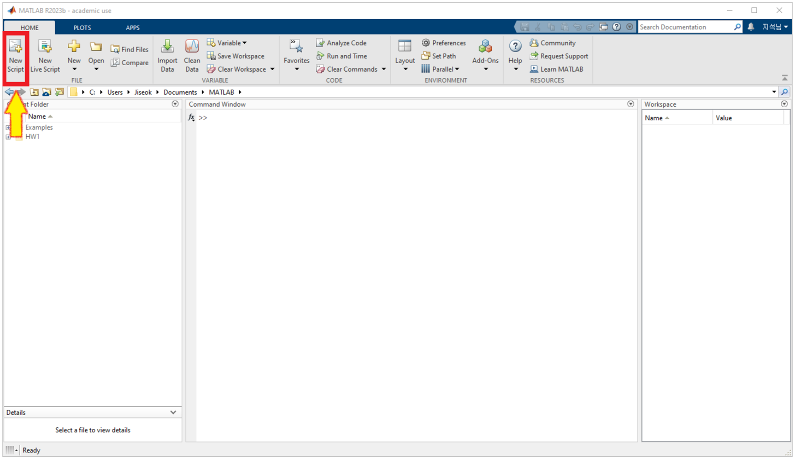
In MATLAB, you can write an `.m` file.

An `.m` file is either...

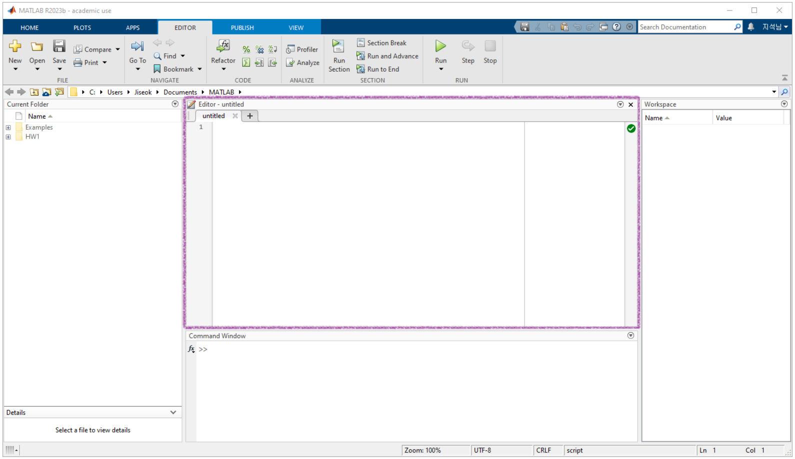
- a script, which is just a collection of commands, or
- a function file, which can be used to define a function.

Let's first see what scripts are.

To make an `.m` file, click the [Home] > [New Script] button.



A new window named Editor will appear.



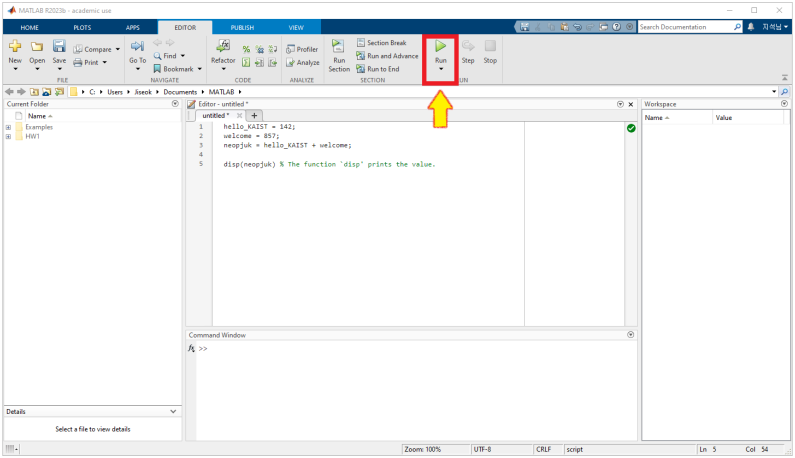
Let us make an `.m` file which does something we have done last week.

To indicate that we are working on the Editor, we will shade the background. Type in the following commands in the Editor.

```
hello_KAIST = 142;
welcome = 857;
neopjuk = hello_KAIST + welcome;

disp(neopjuk) % The function ‘disp’ prints the value.
```

To run the `.m` file we have just wrote, press the `F5` key in the keyboard, or click button [Editor] > [Run].



```
hello_KAIST = 142;
welcome = 857;
neopjuk = hello_KAIST + welcome;

disp(neopjuk) % The function `disp' prints the value.
```

```
>>
```

To run the `.m` file we have just wrote, press the `F5` key in the keyboard, or click button [Editor] > [Run].

Before you actually run the code, MATLAB will force you to save the `.m` file. Let us save it with the name `hello.m`.

The result:

hello.m

```
hello_KAIST = 142;
welcome = 857;
neopjuk = hello_KAIST + welcome;

disp(neopjuk) % The function ‘disp’ prints the value.
```

```
>> hello
   999

>>
```

Contents

Recall that `.m` files have two roles: it is either...

- a script, which is just a collection of commands, or
- a function file, which can be used to define a function.

Now let's look at the second usage.

Let us start with a basic example.

Suppose we want to make a function `triple` which triples the input.

The basic template for defining functions is:
`function` $\langle \textit{output} \rangle = \langle \textit{function name} \rangle (\langle \textit{inputs} \rangle)$

We make an `.m` file as below.

The file name MUST be the same with the function name.

triple.m

```
function res = triple(x)
res = 3 * x;
```

Save the file `triple.m`. You *should* be able to see the file `triple.m` in the Current Folder panel on the left of the Editor window.

Once you saved the file, you can use the function `triple` as any other functions you have seen so far.

```
>> triple(4)

ans =

    12

>> triple([2, 3])

ans =

     6     9

```

A function can take multiple inputs.

axpy.m

```
function res = axpy(a, x, y) % ``a x plus y''
res = a * x + y;
```

```
>> axpy(4, 7, 9)

ans =

    37
```

If you are keep working on the same folder, then you will still have `triple.m` in the Current Folder. Then you can even do something like this:

actually_double.m

```
function res = actually_double(x)
res = triple(x);
res = res - x;
```

```
>> actually_double(5)

ans =

    10
```

Why do we use the name `actually_double` instead of just `double`?

This is because there already exists a predefined (built-in) MATLAB function `double`.

Built-in functions have higher priority than user-defined functions.

Since the built-in `double` is not a function which doubles the input, we will see unexpected behaviors.

Functions also can have multiple outputs.

head_tail.m

```
function [H, T] = head_tail(x)  % Use square brackets!
H = x(1);
n = numel(x);  % number of elements in x
T = x(n);
```

```
>> x = [2, 3, 5, 7, 11];
>> head_tail(x)

ans =

     2    % ?!?!
```

The automatically generated variable `ans` only stores the first output. But we know what to do.

```
>> x = [2, 3, 5, 7, 11];
>> [h, t] = head_tail(x)

h =

     2


t =

    11
```

You might forget what your function does. You can add a description as follows.

head_tail.m

```
function [H, T] = head_tail(x)

% Takes a vector x. Returns H and T,
% where H is the first element of x
% and T is the last element of x.

H = x(1);
n = numel(x);
T = x(n);
```

After modifying `head_tail.m` as the previous slide, you can use the `help` command to see the description you wrote.

```
>> help head_tail
Takes a vector x. Returns H and T,
where H is the first element of x
and T is the last element of x.
```

You can use `help` to see descriptions of built-in functions and variables. For example,

```
>> help eye
```

will show you a description on the built-in function `eye`, and

```
>> help pi
```

will show you a description on the predefined variable `pi`.

Thank you!