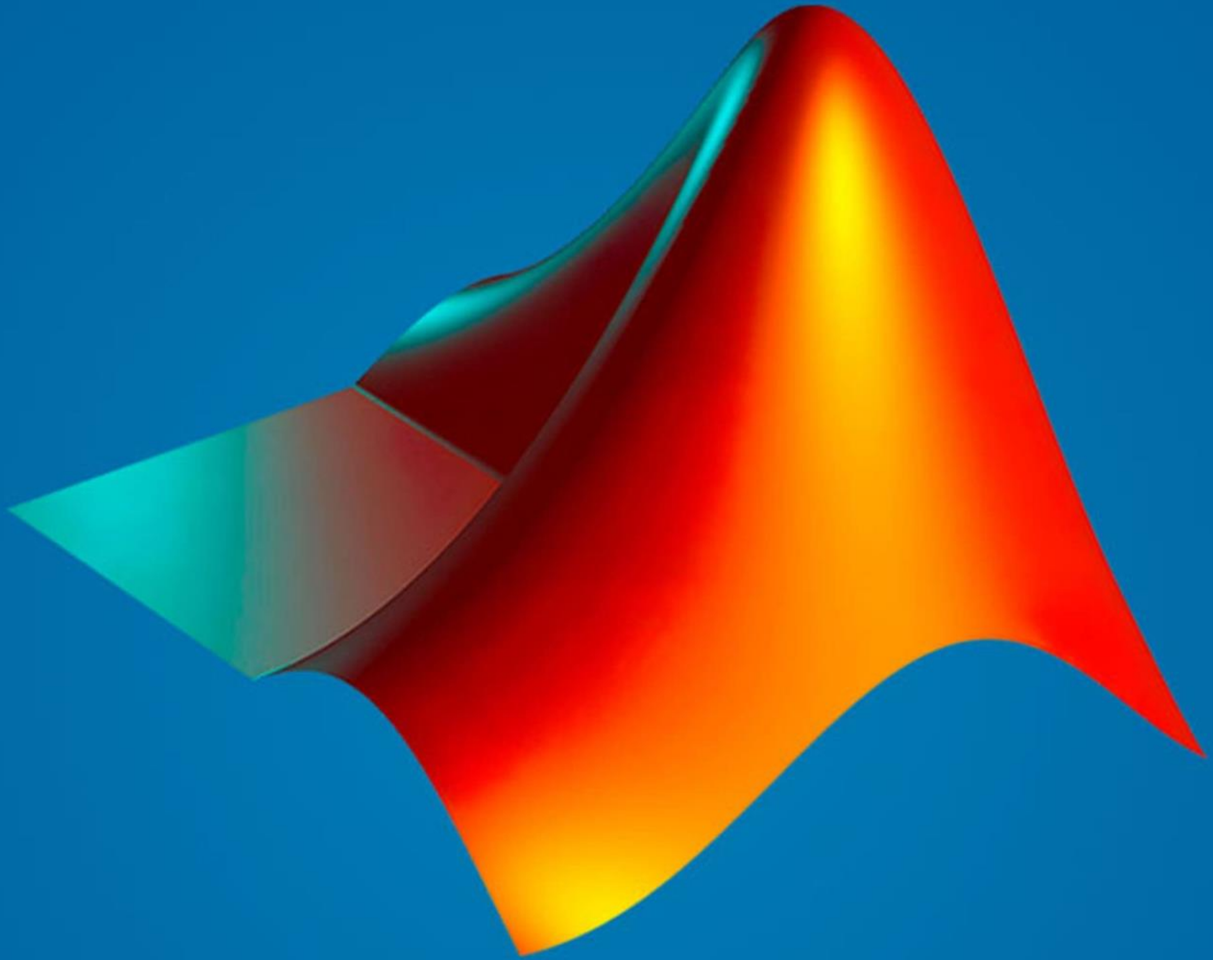
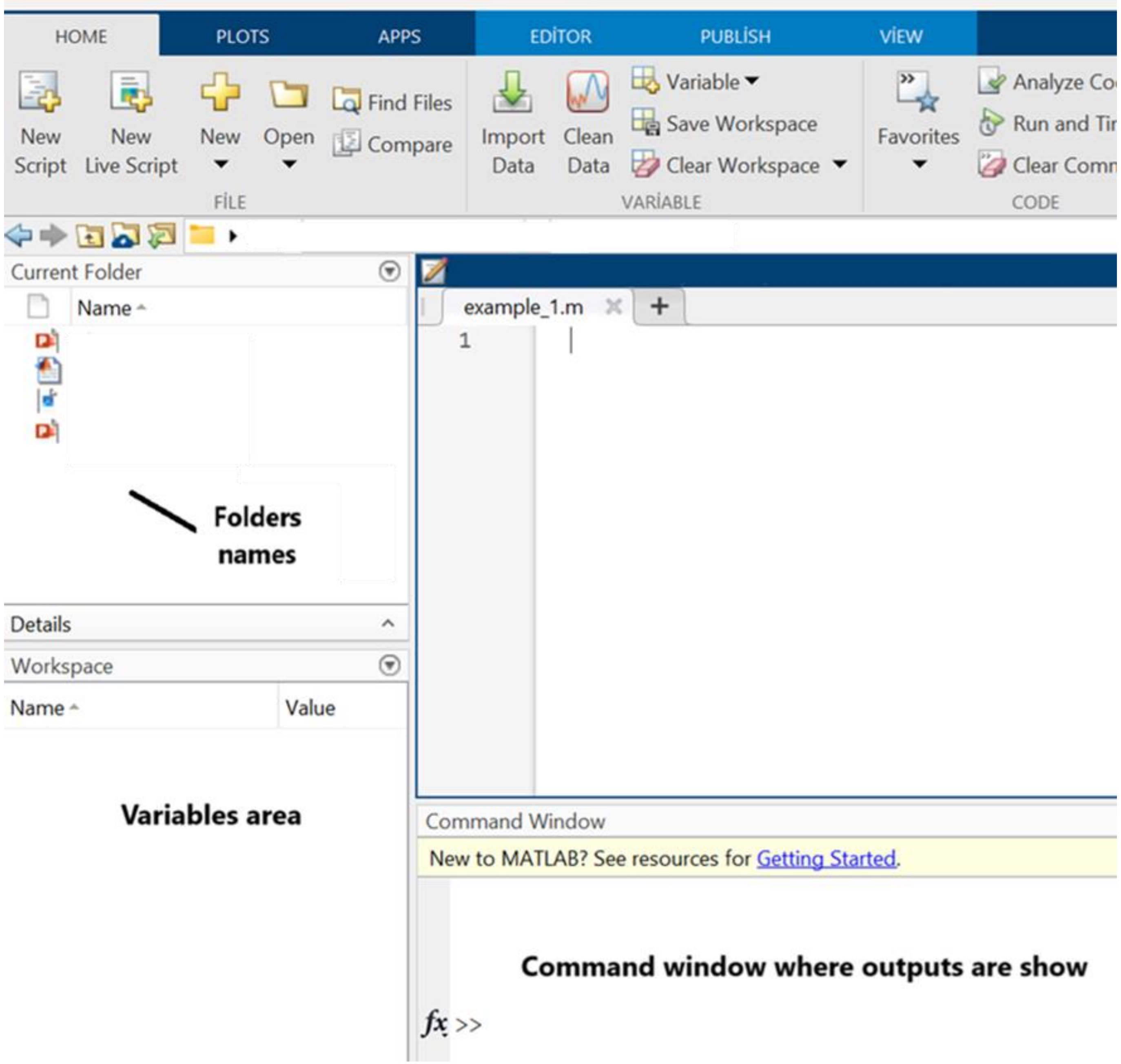


MATLAB

INTRODUCTION

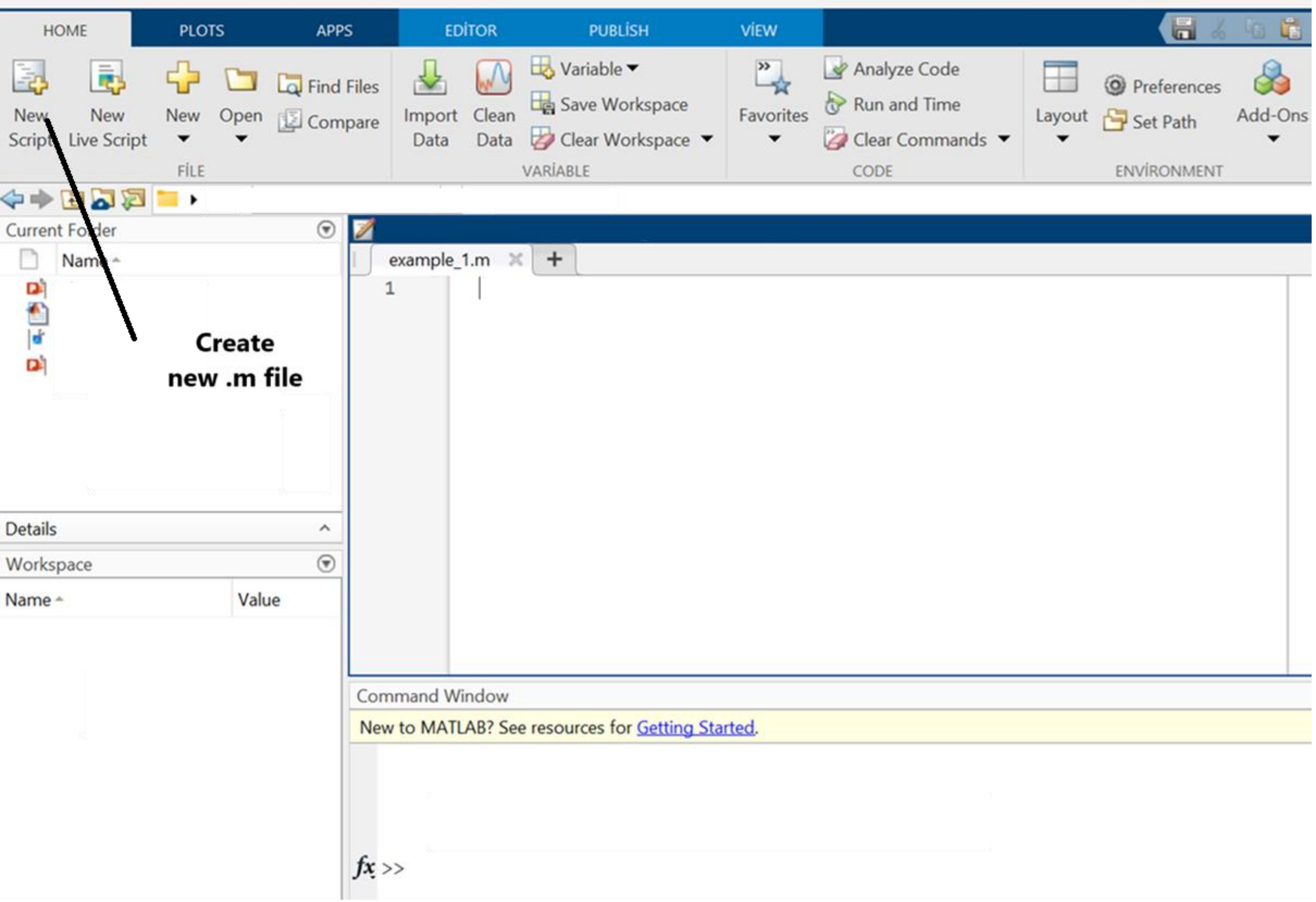
- MATLAB (matrix laboratory) is a fourth-generation high-level programming language.
- It allows:
  - ➢Signal Processing and Communications
  - ➢Image and Video Processing
  - ➢Control Systems
  - ➢Test and Measurement

➢MathWorks has many toolbox for different applications.



- This is the main screen of MATLAB. You can adjust it as you wish to by using drag and drop.
- For instance, If I want to change the place of workspace, hold down the left click, drag and drop it.

- In file place, you can access all files in the current folder.
- In workspace, variable names and their contents are shown.
- Command window is the area where the outputs appear.

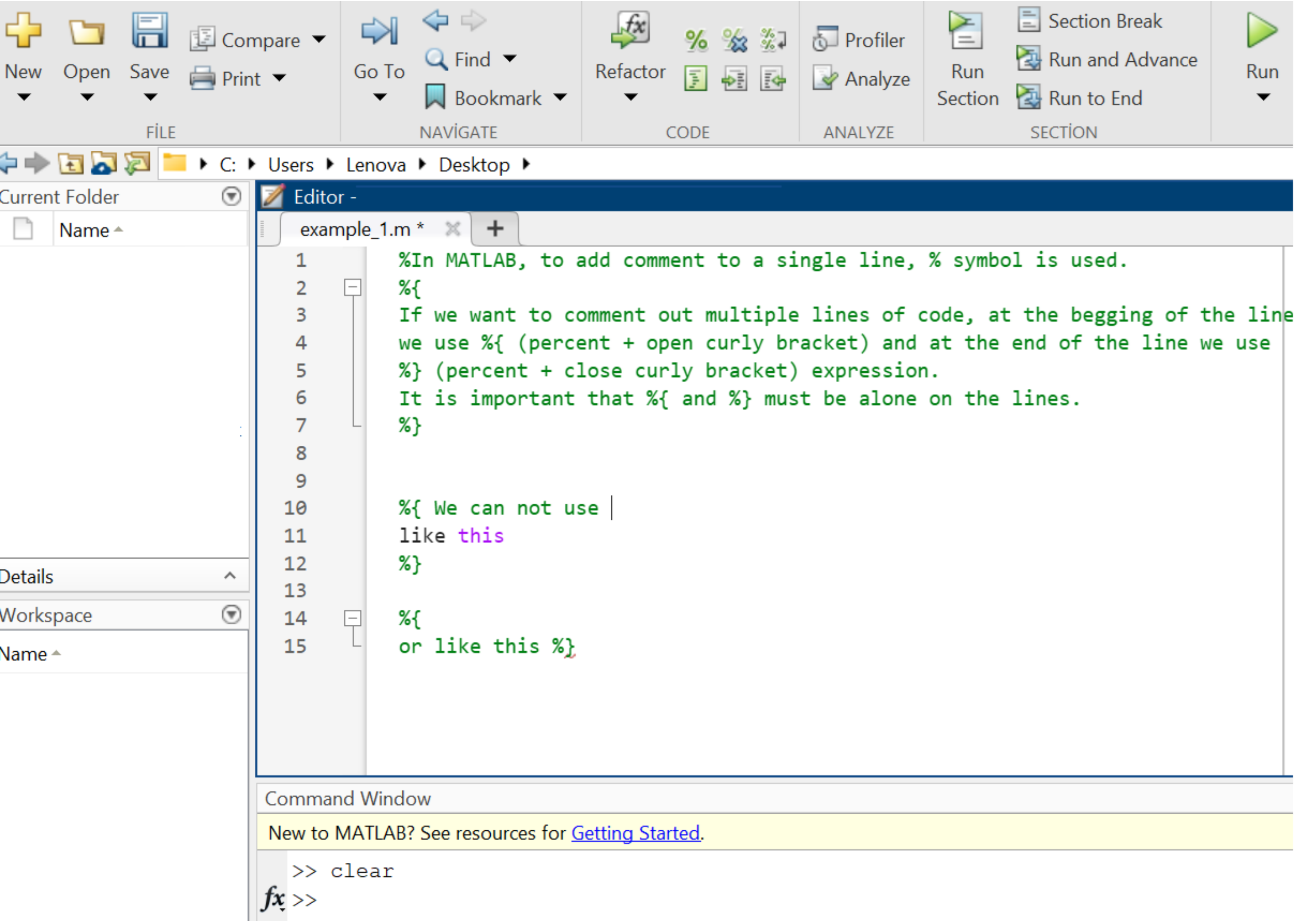


Create new .m file

New to MATLAB? See resources for Getting Started.

```
fx >>
```

- Let's create a new MATLAB file and write our first code.
- From the new script, we can create a new MATLAB file.
- When we do cltr + save, we can save the .m file anywhere we want to. (I named the my .m file as example_1, you can name it as you wish)



```
%In MATLAB, to add comment to a single line, % symbol is used.
%{
If we want to comment out multiple lines of code, at the begging of the line
we use %{ (percent + open curly bracket) and at the end of the line we use
%} (percent + close curly bracket) expression.
It is important that %{ and %} must be alone on the lines.
%}


%{ We can not use
like this
%}

%{
or like this %}
```

New to MATLAB? See resources for Getting Started.

```
>> clear
>>
```

In MATLAB, to add comment to a single line, % symbol is used.

If we want to comment out multiple lines of code, at the begging of the line we use %{ (percent + open curly bracket) and at the end of the line we use %} (percent + close curly bracket). It is important that %{ and %} must be alone on the lines.

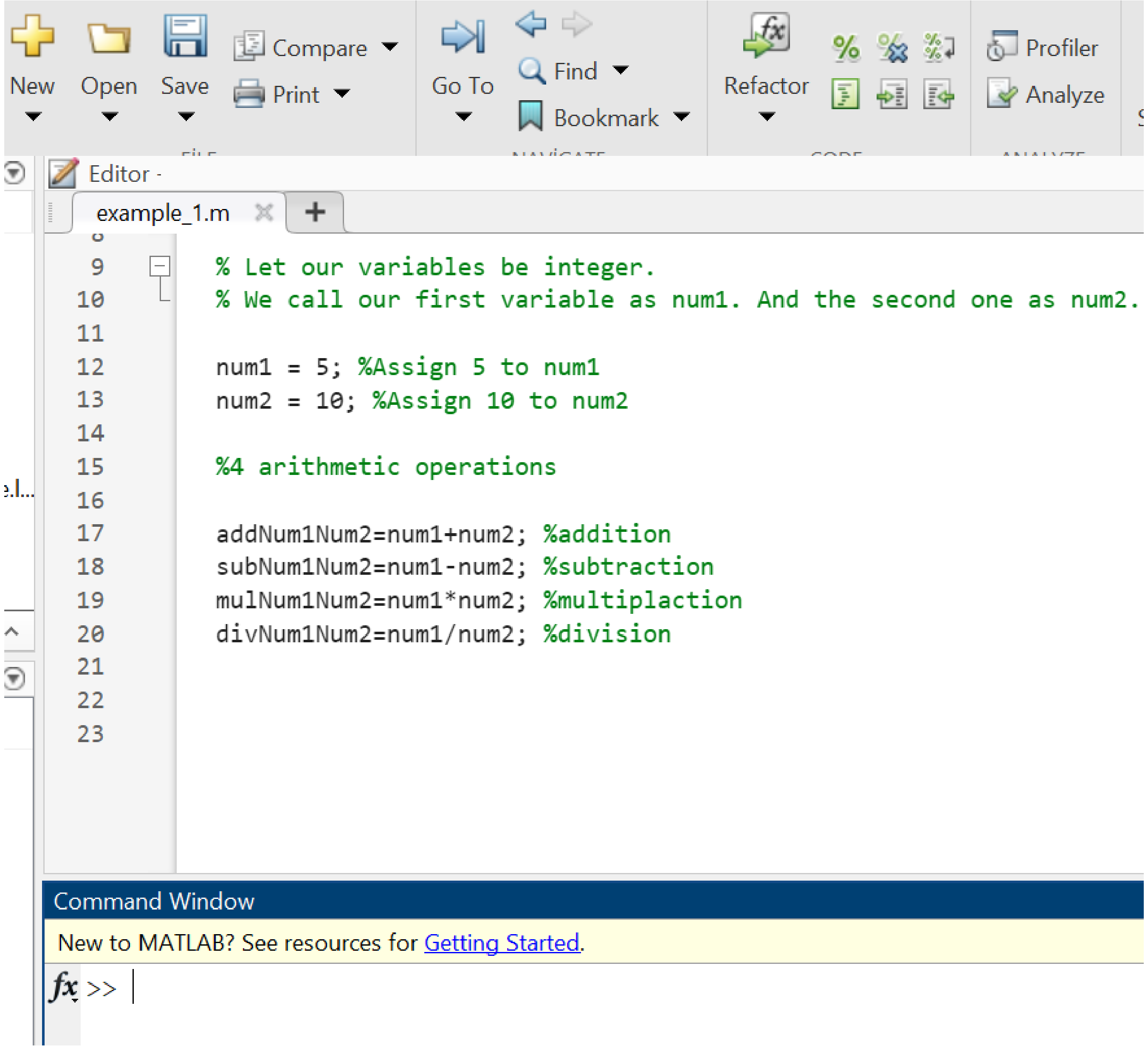
%{ We can not use

like this

%}

%{

or like this %}

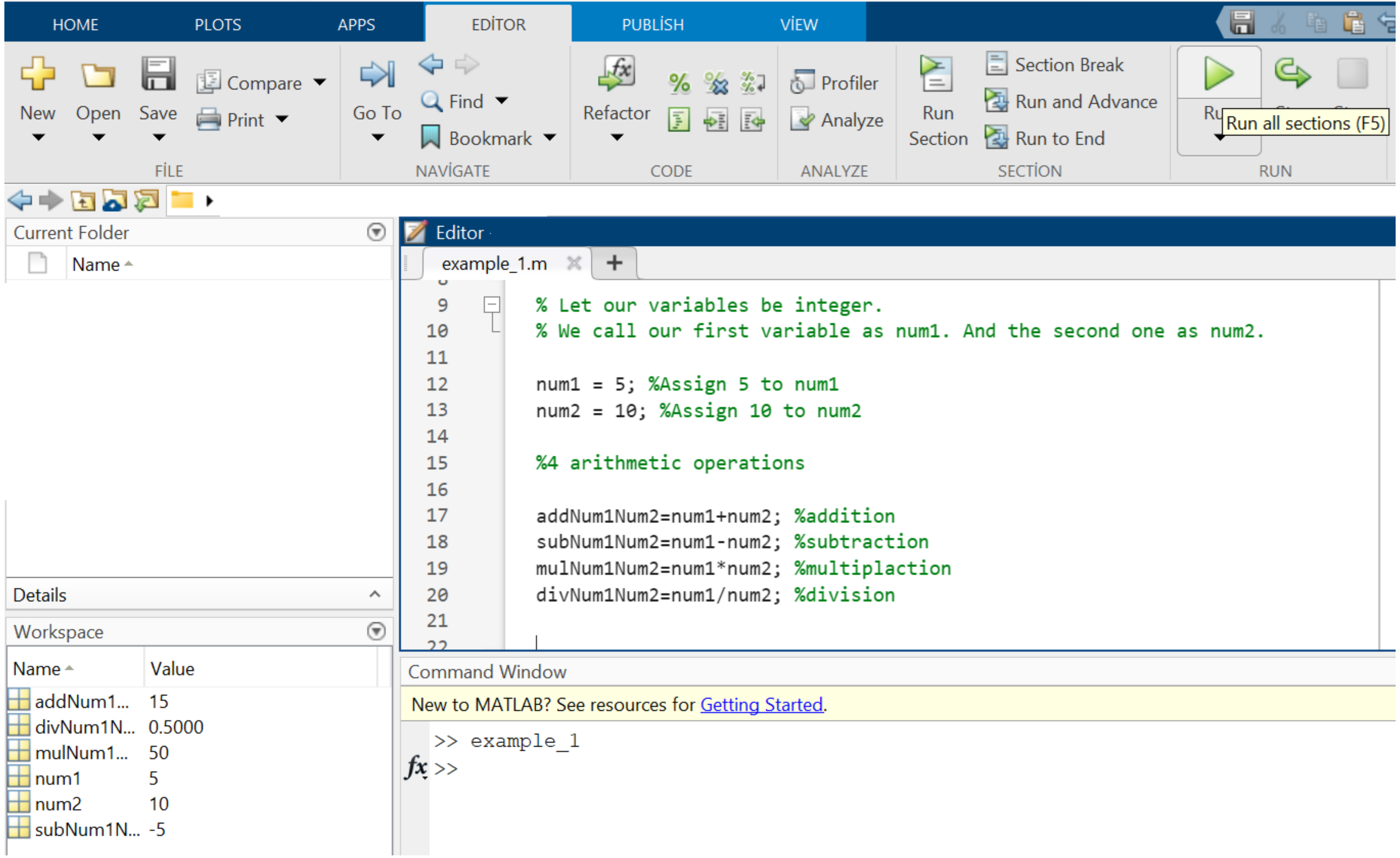


```
% Let our variables be integer.
% We call our first variable as num1. And the second one as num2.

num1 = 5; %Assign 5 to num1
num2 = 10; %Assign 10 to num2

%4 arithmetic operations

addNum1Num2=num1+num2; %addition
subNum1Num2=num1-num2; %subtraction
mulNum1Num2=num1*num2; %multiplaction
divNum1Num2=num1/num2; %division
```

New to MATLAB? See resources for Getting Started.

```
>>
```

- Now let’s create variables and assign value to them.
- Let our variables be integer. We call our first variable as num1. And the second one as num2.
- For these variables, perform 4 arithmetic operations which are addition, subtraction, multiplication and division.



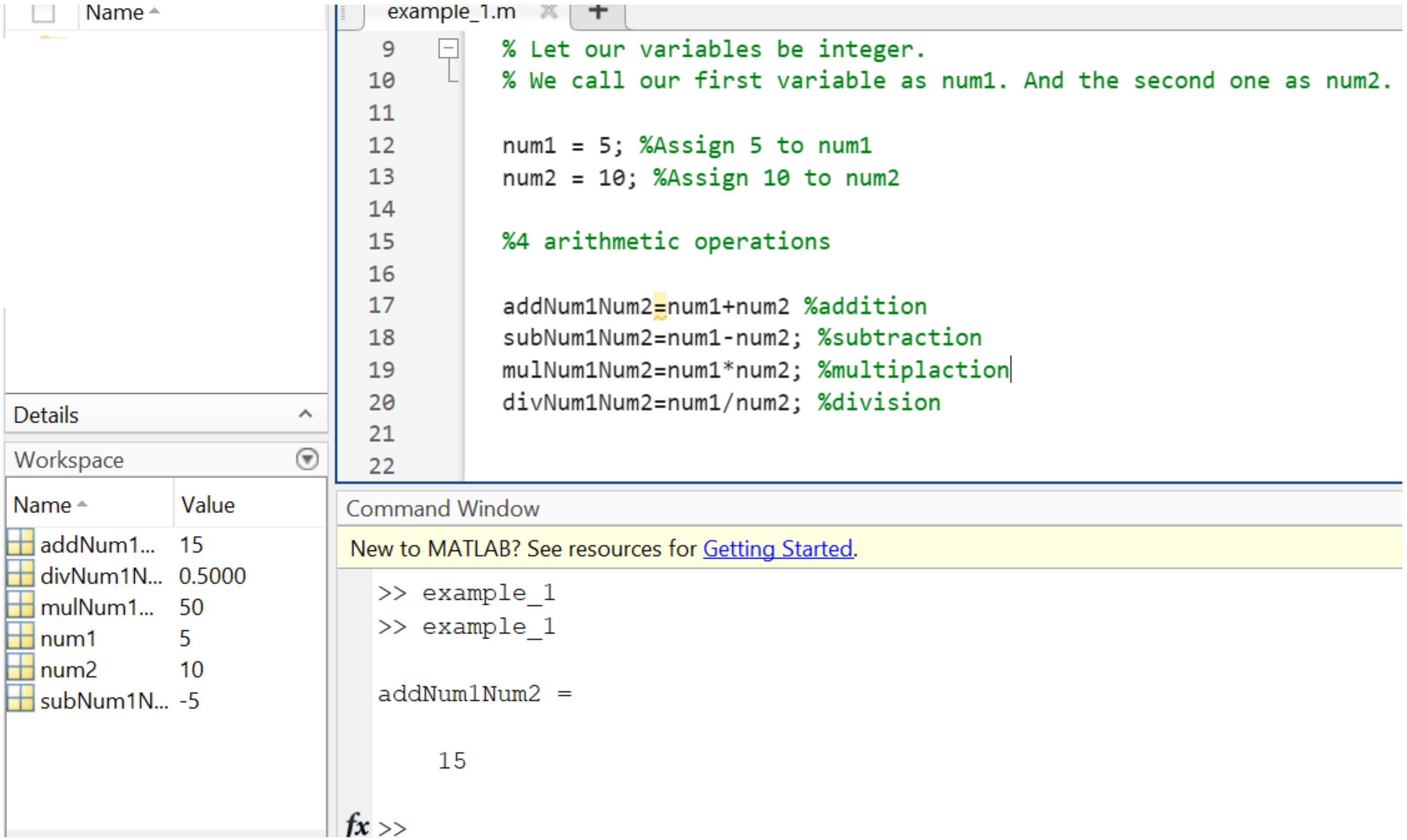
```
% Let our variables be integer.
% We call our first variable as num1. And the second one as num2.

num1 = 5; %Assign 5 to num1
num2 = 10; %Assign 10 to num2

%4 arithmetic operations

addNum1Num2=num1+num2; %addition
subNum1Num2=num1-num2; %subtraction
mulNum1Num2=num1*num2; %multiplaction
divNum1Num2=num1/num2; %division
```

Name	Value
addNum1...	15
divNum1N...	0.5000
mulNum1...	50
num1	5
num2	10
subNum1N...	-5

New to MATLAB? See resources for Getting Started.

```
>> example_1
>>
```

- To exacute our code, in the editor tab click on RUN. Or as a shortcut you can use the f5.
- In MATLAB, if we do not put a semicolon (;) at the end of the line, in command window the content of this line will appear.
- As you can see, There is no output on the command window.



```
 9  % Let our variables be integer.
10  % We call our first variable as num1. And the second one as num2.
11
12  num1 = 5; %Assign 5 to num1
13  num2 = 10; %Assign 10 to num2
14
15  %4 arithmetic operations
16
17  addNum1Num2=num1+num2 %addition
18  subNum1Num2=num1-num2; %subtraction
19  mulNum1Num2=num1*num2; %multiplaction
20  divNum1Num2=num1/num2; %division
21
22
```

Name	Value
addNum1...	15
divNum1N...	0.5000
mulNum1...	50
num1	5
num2	10
subNum1N...	-5

New to MATLAB? See resources for Getting Started.

```
>> example_1
>> example_1

addNum1Num2 =

    15

fx >>
```

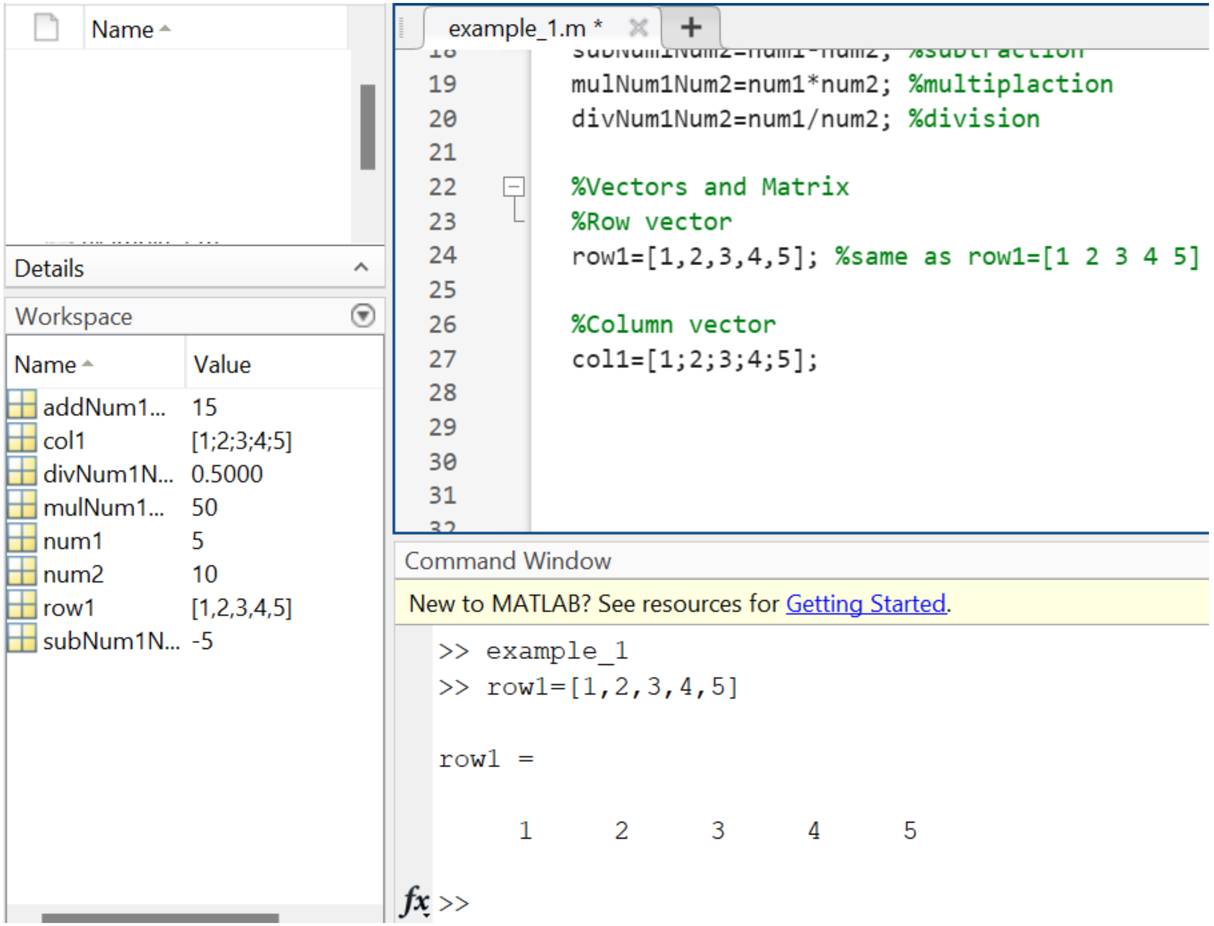
- For instance, we do not put a semicolon (;) at the end of the 17th line.

- When exacuted the code, in command window the content of this line appeared.

- You can see the contents of the variables from the workspace.

- In workspace, when click on the variable name, the content of this variable will open.

- We can write code directly to command window.
- For instance, when we want to clean the command window, type “clc” (without quotation marks, of course) and the command window will be empty.
- Also, we can delete all variables from the workspace. For this “clear” is used. When we type "clear", all variables are deleted.
- Also, we can delete a specific variable from the workspace. For instance, when we write “clear num1”, num1 variable will be deleted.
- Generally, these two expression (clc and clear) are used at the beginning of code.



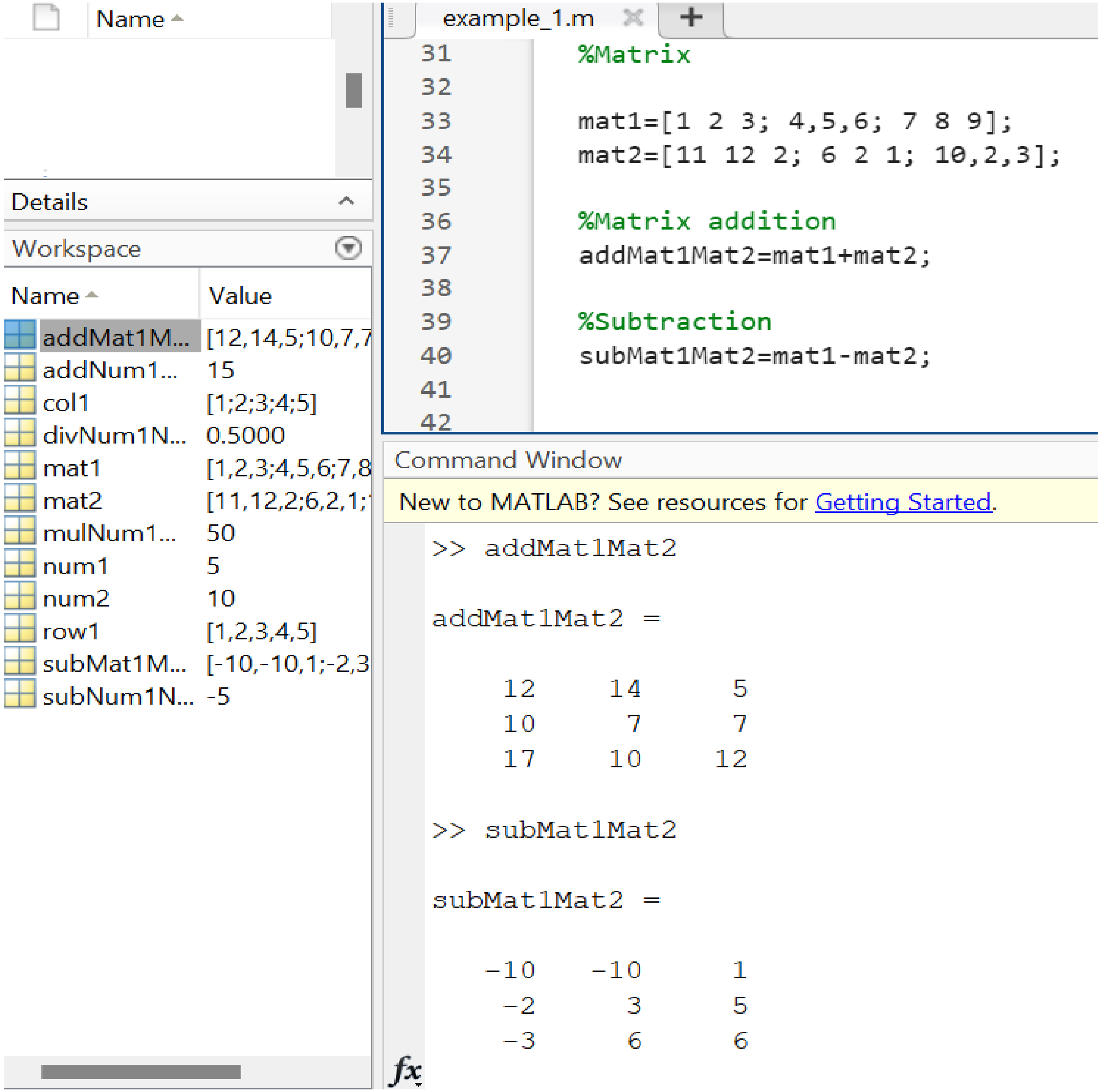
```
subNum1Num2=num1-num2; %subtraction
mulNum1Num2=num1*num2; %multiplaction
divNum1Num2=num1/num2; %division

%Vectors and Matrix
%Row vector
row1=[1,2,3,4,5]; %same as row1=[1 2 3 4 5]

%Column vector
col1=[1;2;3;4;5];
```

Name	Value
addNum1...	15
col1	[1;2;3;4;5]
divNum1N...	0.5000
mulNum1...	50
num1	5
num2	10
row1	[1,2,3,4,5]
subNum1N...	-5

New to MATLAB? See resources for Getting Started.

```
>> example_1
>> row1=[1,2,3,4,5]

row1 =

     1     2     3     4     5

fx >>
```

- Now let's define vectors and matrices.
- A vector is a one-dimensional array, which means we can create either row array or column array.
- In row, space or comma(,) are used to separate values.
- As a separator semicolon(;) is utilized for column vector.



Name	Value
addMat1M...	[12,14,5;10,7,7
addNum1...	15
col1	[1;2;3;4;5]
divNum1N...	0.5000
mat1	[1,2,3;4,5,6;7,8
mat2	[11,12,2;6,2,1;
mulNum1...	50
num1	5
num2	10
row1	[1,2,3,4,5]
subMat1M...	[-10,-10,1;-2,3
subNum1N...	-5

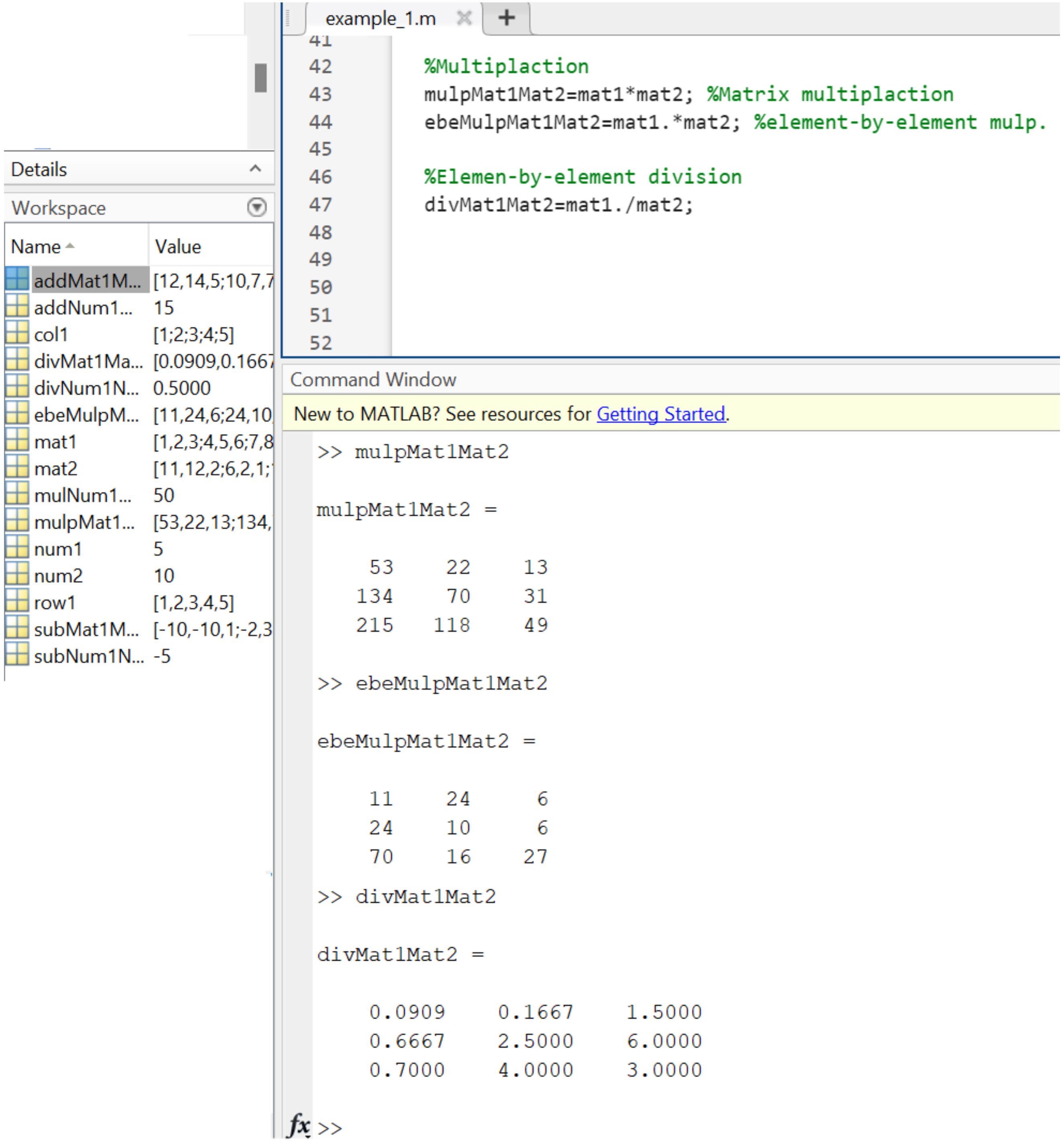
```
%Matrix

mat1=[1 2 3; 4,5,6; 7 8 9];
mat2=[11 12 2; 6 2 1; 10,2,3];

%Matrix addition
addMat1Mat2=mat1+mat2;

%Subtraction
subMat1Mat2=mat1-mat2;
```

New to MATLAB? See resources for Getting Started.

```
>> addMat1Mat2

addMat1Mat2 =

    12    14     5
    10     7     7
    17    10    12

>> subMat1Mat2

subMat1Mat2 =

   -10   -10     1
    -2     3     5
    -3     6     6
```

- Now let’s create matrix and apply matrix operations.

- Firstly, when we do addition and subtraction, as can be seen from the figure MATLAB will do it element by element.



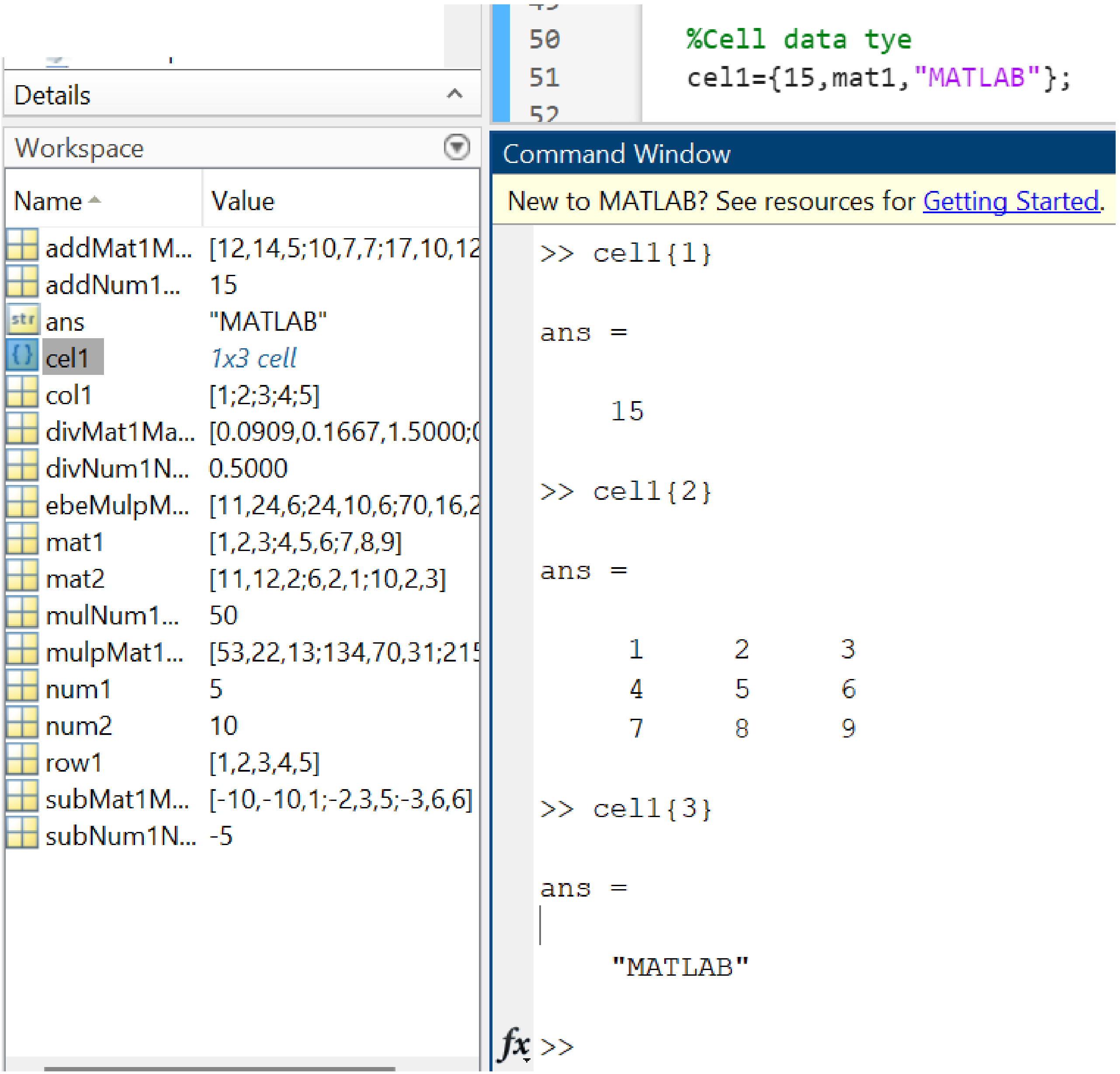
```
%Multiplaction
mulpMat1Mat2=mat1*mat2; %Matrix multiplaction
ebeMulpMat1Mat2=mat1.*mat2; %element-by-element mulp.

%Elemen-by-element division
divMat1Mat2=mat1./mat2;
```

Name	Value
addMat1M...	[12,14,5;10,7,7
addNum1...	15
col1	[1;2;3;4;5]
divMat1Ma...	[0.0909,0.1667
divNum1N...	0.5000
ebeMulpM...	[11,24,6;24,10
mat1	[1,2,3;4,5,6;7,8
mat2	[11,12,2;6,2,1;
mulNum1...	50
mulpMat1...	[53,22,13;134,
num1	5
num2	10
row1	[1,2,3,4,5]
subMat1M...	[-10,-10,1;-2,3
subNum1N...	-5

New to MATLAB? See resources for Getting Started.

```
>> mulpMat1Mat2

mulpMat1Mat2 =

    53    22    13
   134    70    31
   215   118    49

>> ebeMulpMat1Mat2

ebeMulpMat1Mat2 =

    11    24     6
    24    10     6
    70    16    27

>> divMat1Mat2

divMat1Mat2 =

    0.0909    0.1667    1.5000
    0.6667    2.5000    6.0000
    0.7000    4.0000    3.0000

fx >>
```

- For multiplication, If * operator is used, MATLAB will do matrix multiplication.
- To do element-by-element multiplication, .* is used.
- Similarly, ./ is used to make division element-by-element.



Name	Value
addMat1M...	[12,14,5;10,7,7;17,10,12
addNum1...	15
ans	"MATLAB"
cel1	*1x3 cell*
col1	[1;2;3;4;5]
divMat1Ma...	[0.0909,0.1667,1.5000;
divNum1N...	0.5000
ebeMulpM...	[11,24,6;24,10,6;70,16,2
mat1	[1,2,3;4,5,6;7,8,9]
mat2	[11,12,2;6,2,1;10,2,3]
mulNum1...	50
mulpMat1...	[53,22,13;134,70,31;215
num1	5
num2	10
row1	[1,2,3,4,5]
subMat1M...	[-10,-10,1;-2,3,5;-3,6,6]
subNum1N...	-5

```
%Cell data tye
cel1={15,mat1,"MATLAB"};
```

New to MATLAB? See resources for Getting Started.

```
>> cel1{1}

ans =

    15

>> cel1{2}

ans =

     1     2     3
     4     5     6
     7     8     9

>> cel1{3}

ans =

    "MATLAB"

>>
```

- **Cell data type:** Cell array is a data type that can contain any type of data.
- The curly brackets({}) is used to create a cell.
- Create a cell and name it cel1. Let the first element of the cell be an integer, the second one be a matrix, and the third element be a string.
- To access the value of a specific index, enclose the indices in curly parentheses. Such as cel1{1} gives us 15.