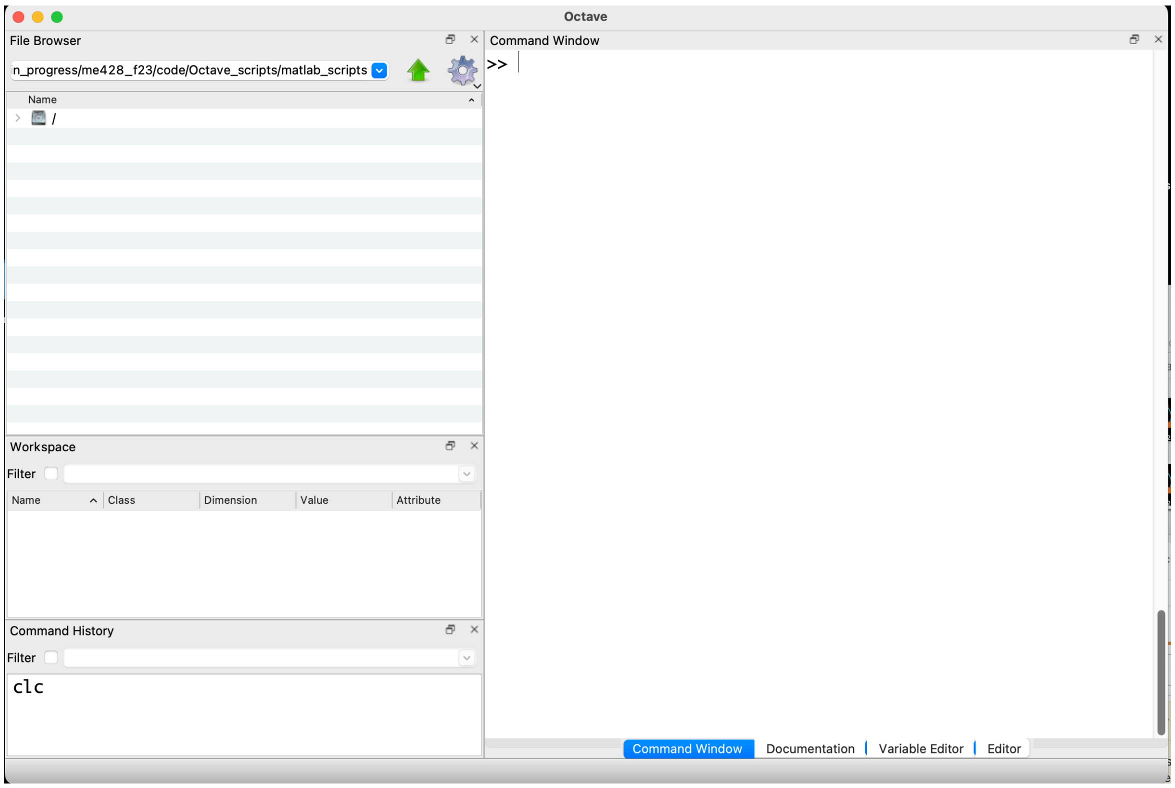
Pranav Bhounsule
Octave Basics. **Using command line**

1. Download and install GNU octave from here: https://octave.org/download

2. Start Octave by navigating through your installed application or click the short cut icon for Octave shown next.

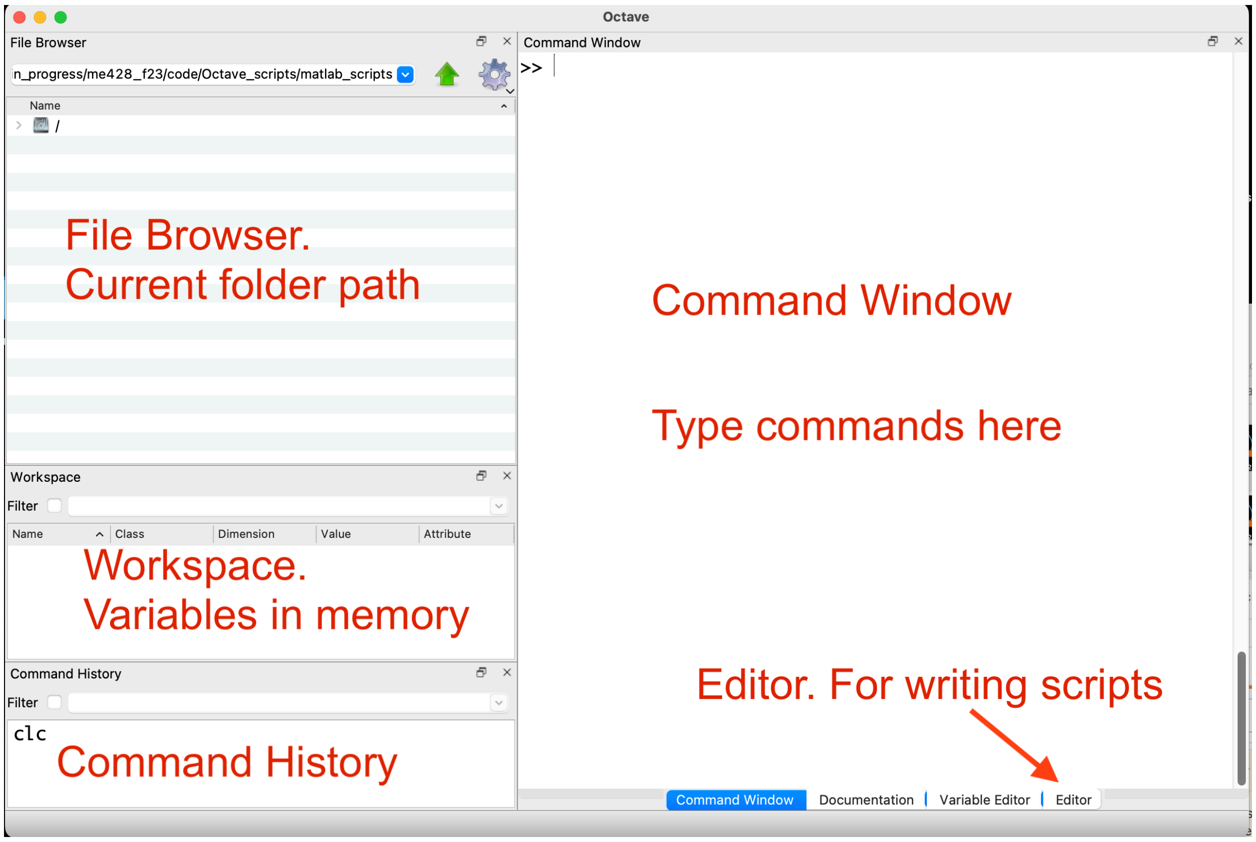
2. You will see the following screen once Octave loads up.



```
clc
```

```
>>
```

3. The different panels are marked below

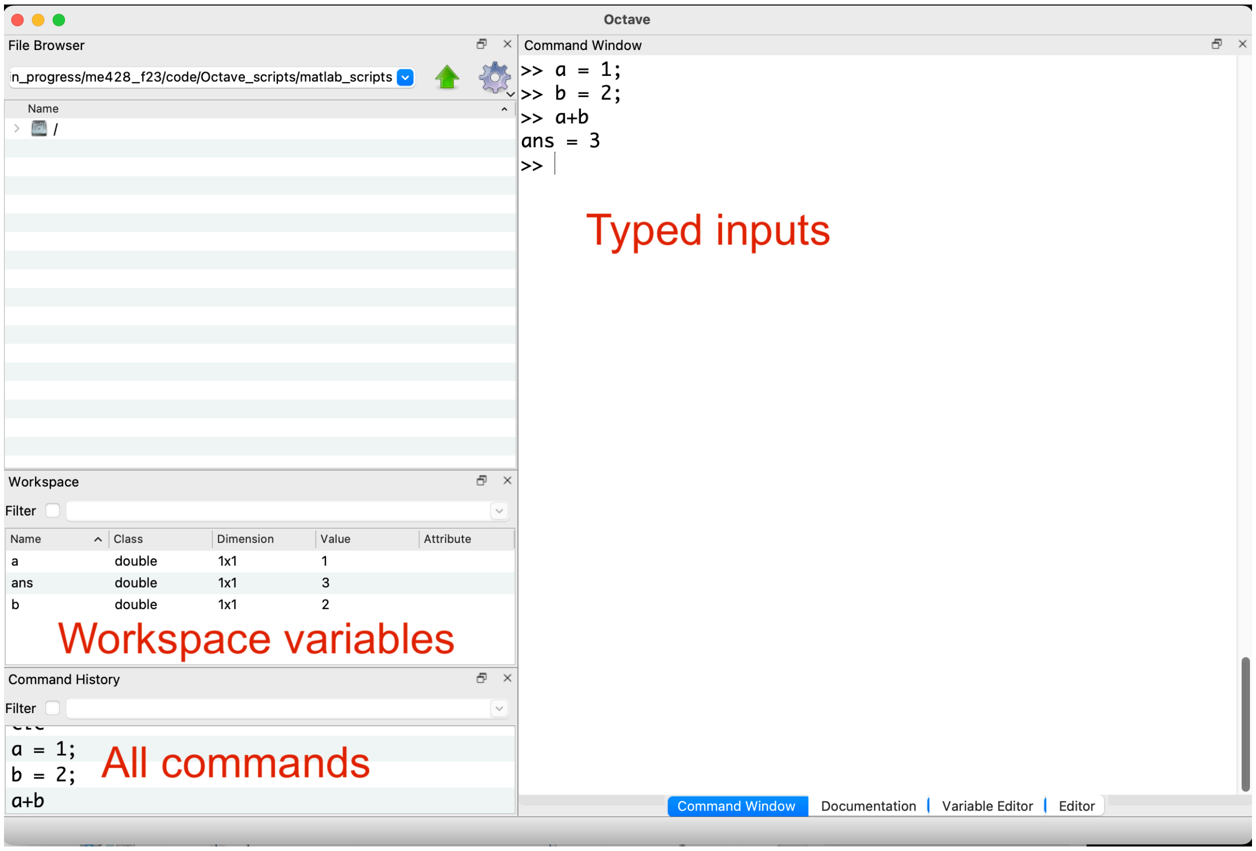


Note the Command Window has the symbol >>. This is where you will type. After typing you press ENTER to register the command.

4. **Using Octave as a calculator**

```
>> a = 1;
>> b = 2;
>> a+b
```

The output is shown below.



```
>> a = 1;
>> b = 2;
>> a+b
ans = 3
>>
```

Name	Class	Dimension	Value	Attribute
a	double	1x1	1	
ans	double	1x1	3	
b	double	1x1	2	

```
a = 1;
b = 2;
a+b
```

NOTE a couple of things

I. The variables in memory appear in the workspace. (left-middle panel)

II. The commands also appear in the Command History (left-bottom panel)

III. The sum a+b is printed as 3.

IV. Putting a semi colon suppresses the output. For example, if we put a=1 then we would get something like a = 1 displayed on the next line. Try it out. Similarly if we have put a+b; then we would not have got the answer.

V. We could also do something like

```
>> c = a+b;
>> c
```

This would lead to something like

```
>> c =
        3
```

From here on, I will only put things we type in the Command Window and the output.
5. **Vectors in different ways**

```
>> d = [1 2 3] %This is comment.
```

NOTES: Comments start with %. What you type after % is ignored by Octave. The above line produces the row matrix, i.e. one row and three column.

```
>> e = [1 2 3]’
```
This produces the column matrix, i.e. one column and three rows. The apostrophe here does the transpose of the matrix

Another way of writing the same matrix is
```
>> e = [1; 2; 3]
```

6. **All inputs to Octave are matrices**

You can check the number of rows and columns by typing invoking the function size.

Try the following
```
>> size(a)
```
The answer will be 1 1, indicating this is a matrix with 1 row and 1 column.

Try these out.
```
>> size(d)
>> size(e)
```

7. **Using help in Octave**
If you need help on something just type help ‘command name’

```
>> help size
```

8. **Generate random matrices.**
Let us generate a random matrix of size 3 by 3.

```
>> A = rand(3,3)
```
This tells Octave to generate a random matrix of size 3 by 3. For more info you can type help rand

If you want the element in Row 1 and Column 1 you need to type A(1,1). For the element in Row 1 and Column 2 you type A(1,2).

9. **Finding the inverse**

```
>> B = inv(A)
```

To check the result try

```
>>A*B
```

As you know A into its own inverse will give the identity matrix.

10. **Generating a list of numbers**

```
>> t = linspace(0,6,100);
```

This generates the matrix t with 1 row and 100 columns in the range from 0 to 6.

11. **Basic Plotting**

```
>> x = sin(t)
```

x is also going to be a matrix with 1 row and 100 columns.

```
>> plot(t,x)
```

This generates a plot of x as a function of t

```
>> xlabel(‘x’); ylabel(’t’)
```

This puts labels on the x and y axis.

>> You can customize the plot. For e.g. if you want the plot to have x instead of continuous line try

```
>> plot(t,x,’rx’);
>> figure(1)
```

12. **Multiple Plots can be done using subplot.** Try these commands

```
>> subplot(2,1,1); plot(t,x,’rx’);
>> subplot(2,1,2); plot(t,y,’bo’);
>> figure(1)
```

13. **Extra (do later)**

You will learn best by playing with Octave. Try these commands

```
>>eye(3)
>>ones(4,2)
>> clc
>>zeros(2,3)
>> clear all
```

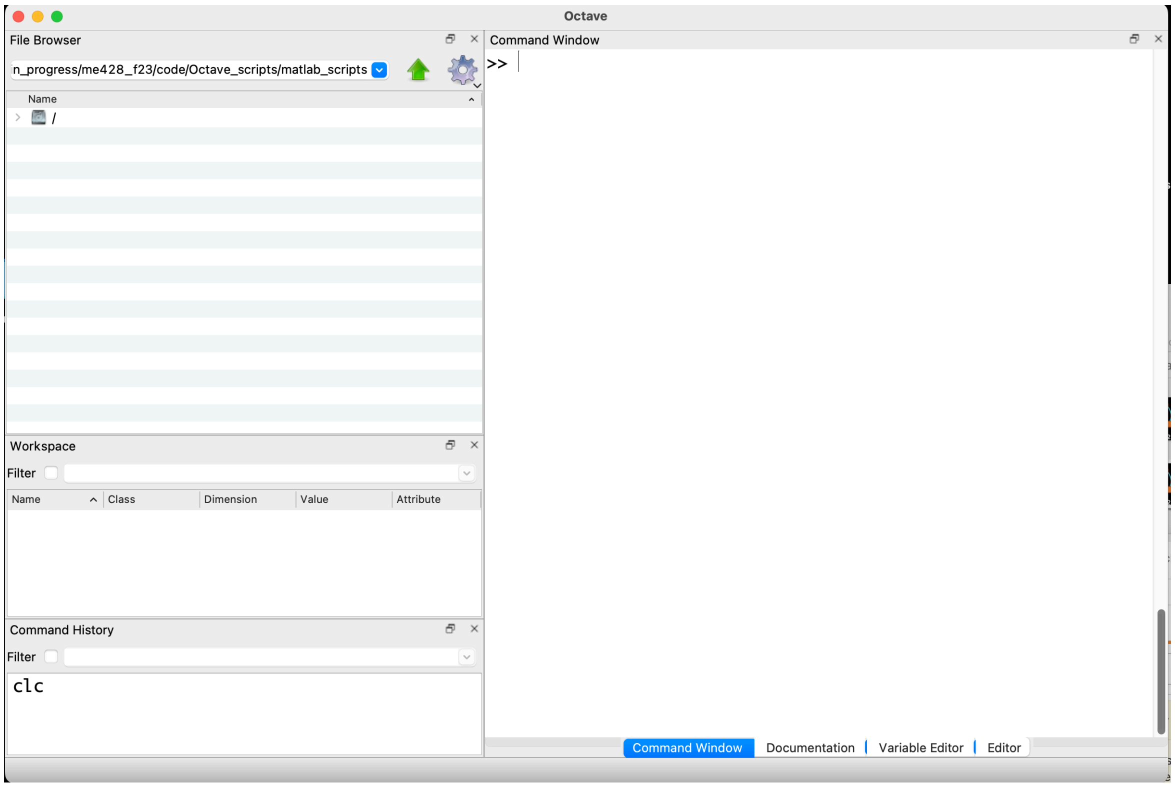
The last command will clear variables in the workspace (see 2nd figure in this workout to locate the Workspace).

To learn more about a function just type help <function_name> at the command prompt.

Pranav Bhounsule
Octave Scripts using editor to write functions.

1. Start Octave by navigating through your installed application or click the short cut icon for Octave shown next.

2. You will see the following screen once Octave loads up.

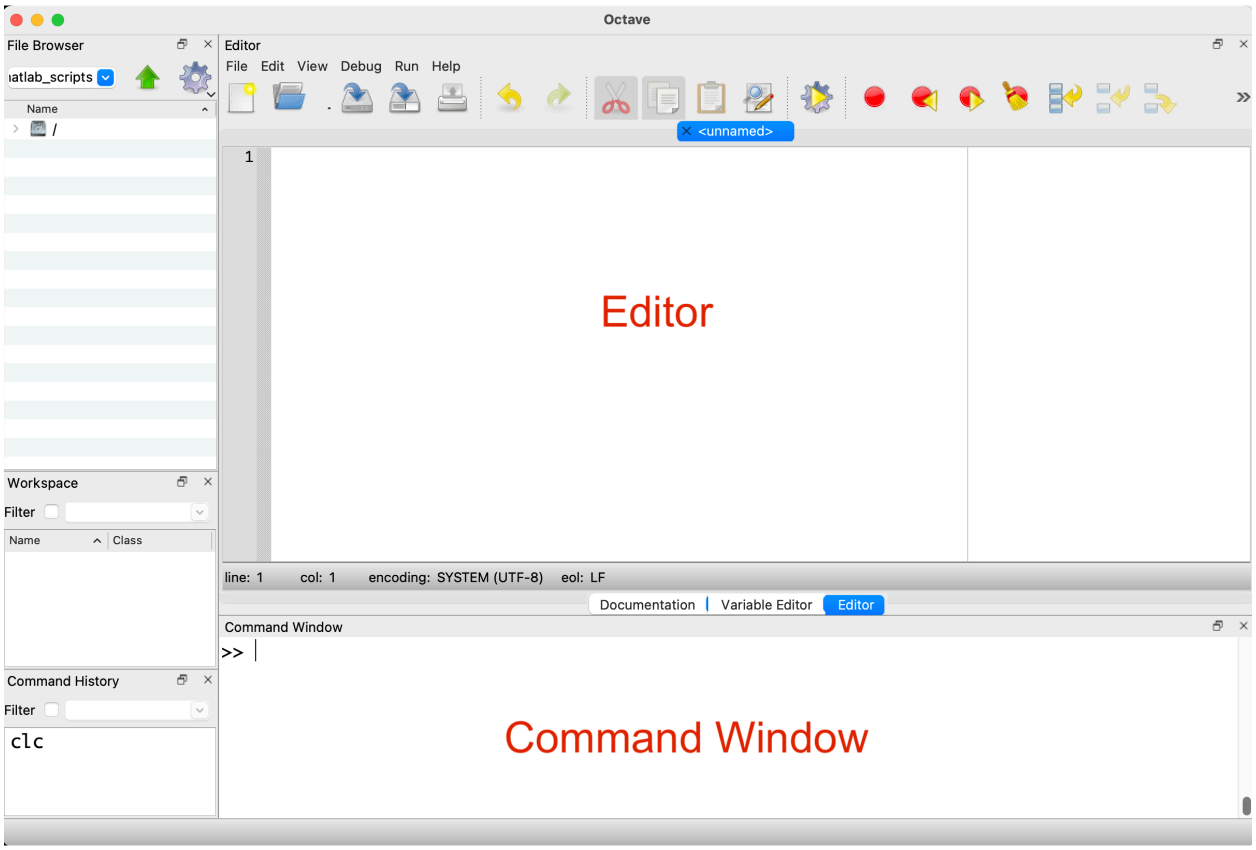


```
>>
```

```
clc
```

3. We will write a script file. To do this go to File > New Script.

(This step is not mandatory but it will help to have this layout for the workflow) Next, follow this video to change the layout to the following:
https://youtu.be/0USOvYHLqSU



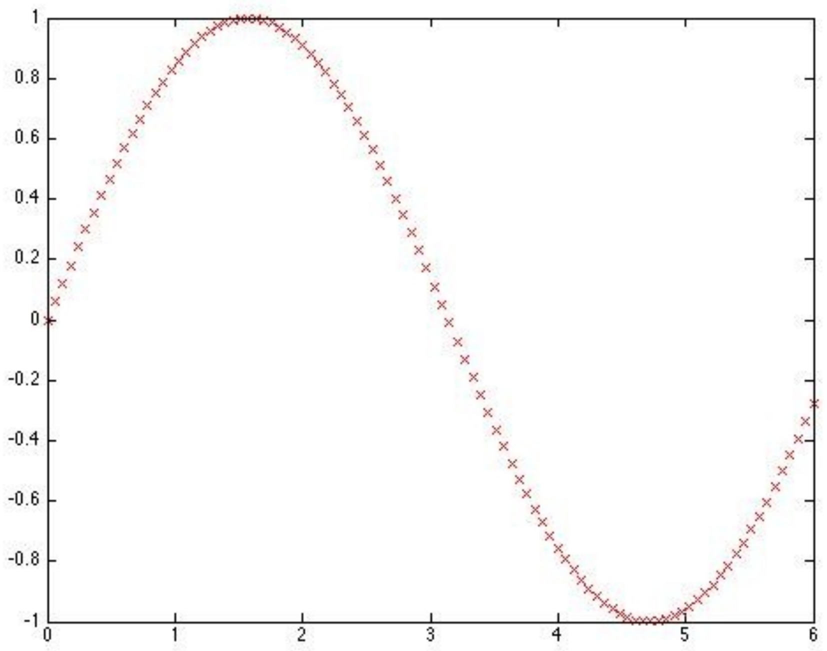
4. **Generating a plot.**
Open a new script file, *New > Script file*.
In the Editor type the following or copy-paste the code below

%%%%%%%%% Copy paste the code below %%%%%%%%%

```
clc %clear screen
clear all %clear all workspace variables.
close all %Close all figures.
x = linspace(0,2*pi); %Generate values of x linearly space in the interval 0 to 6.
y = sin(x); %Generate values of y
plot(x,y,'rx'); %Generate a plot
```

%%%%%%%% Code ends here %%%%%%%%%%%%%%%%%%

Now save the file, *Save -> test1.m*
Next run the file, Run (press the green arrow button in the “Editor tab”)
If everything worked fine then you will see the following figure displayed in Octave



A couple of things about the code above.
The % denotes a comment. Everything after the % is ignored by Octave.
From the comments you will be able to understand what each command does.

For more information you can also type in the Command Window
help plot
help clc

This file test1.m is called a script. The good part of a making a script is that your code is saved and it is easy to make changes.

5. **Finding the sum of elements of a vector**
Our next problem is to find the sum of elements in a matrix.
For e.g. a = [1 1 3 4]; The output of the program should say 9 (=1+1+3+4).

Open a new file by doing New > Script file

Open a new script file, *New > Script file*.

%%%%%%%% Copy paste the code below %%%%%%%%

```
clc %clear the screen
clear all %clear all variables in the memory
a = [1 1 3 4]; %semi-colon suppresses the output.

total_sum = 0; %this variable will save the running sum.

for i=1:length(a) %for loop, length(a) gives the number of elements in a which is 4.
        total_sum = total_sum + a(i); %this command maintains a running sum of all
variables.
end %this tells Octave that for loop has now ended.

disp(total_sum) %this displays the sum. You could also write simply total_sum

%%%%%%% Code ends here %%%%%%%%%%%%%%%%%
```

Now save the file. *Save As > test2.m*
Run the file, Press green arrow in "editor tab"

You should see the output in the Command Window. It should be 9.

Play with the matrix. Try a =[1 2 3 6 3 2 3]; or anything of your choice and see the output

6. **Writing a generic function for summing elements**
Now what if you have to find the sum of elements of multiple matrices of different sizes. For e.g.

```
a = [1 1 3 4];
b = [6 5 7 ];
c = [10 1 5 6 7];
```

One option is to run the above code multiple times with the above matrices. A better way is to write a function that does this. This gets us to writing functions as a script.

Open a new script file *New > Script file* and copy paste the code below.

```
%%%%%%%% Copy paste the code below %%%%%%%%

function total_sum = my_sum(a) %the function name is my_sum. The input is a. The
output is total_sum.

total_sum = 0; %this variable will save the running sum.

for i=1:length(a) %for loop, length(a) gives the number of elements in a which is 4.
        total_sum = total_sum + a(i); %this command maintains a running sum of all
variables.
```

```
end %this tells Octave that for loop has now ended.

%%%%%%% Code ends here %%%%%%%%%%%%%%%%
```

Next save the code.
Note that Octave calls the function *my_sum*. Save this file.

The above function has to be called from another script file. We will define that next.

Open a new script file *New > Script*

```
%%%%%%%% Copy paste the code below %%%%%%%%
a = [1 1 3 4];
b = [6 5 7 ];
c = [10 1 5 6 7];

sum_a = my_sum(a);
sum_b = my_sum(b);
sum_c = my_sum(c);

disp(sum_a)
disp(sum_b)
disp(sum_c)
%%%%%%% Code ends here %%%%%%%%%%%%%%%%
```

Save the file as *test3.m*
Run the file by clicking the green arrow in the “editor tab”

If everything worked fine the output should be.

```
9
18
29
```