

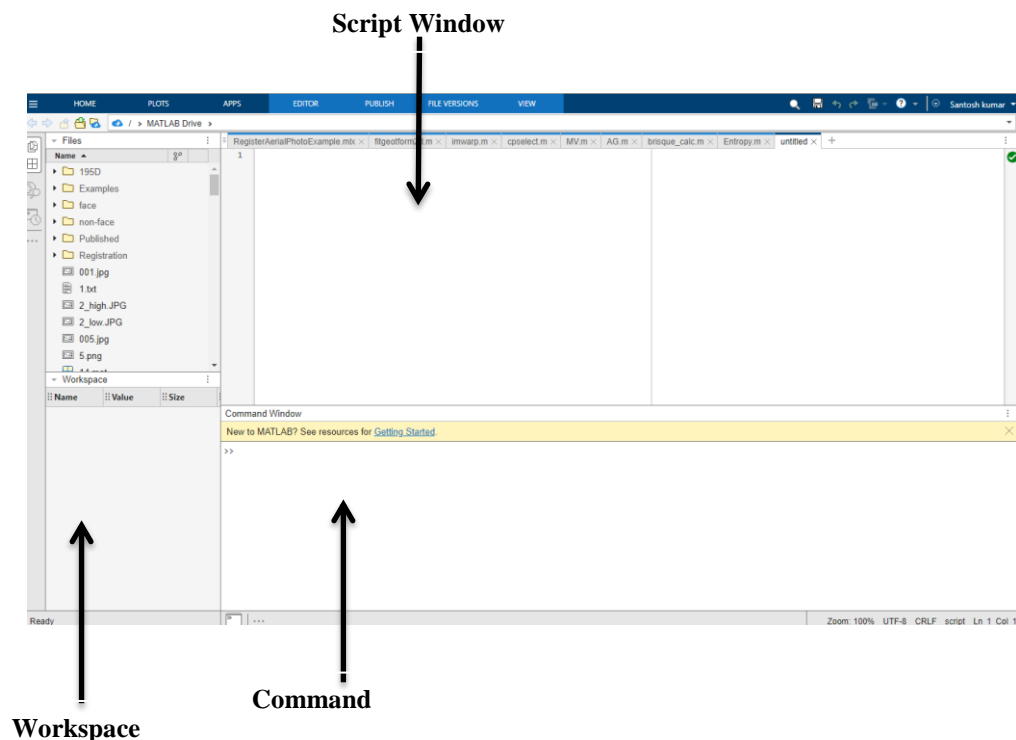
# Introduction to MATLAB

The name MATLAB stands for matrix laboratory. It is a high-performance language for technical computing. It integrates computation, visualization, and programming in an easy-to-use environment where problems and solutions are expressed in familiar mathematical notation. Some of the uses include :

- Math and computation
- Algorithm development
- Data acquisition
- Modeling, simulation, and prototyping
- Data analysis, exploration, and visualization
- Scientific and engineering graphics
- Application development, including graphical user interface building

## MATLAB Environment

- 1) **Command Window** : Use the Command Window to enter variables and to run functions and M-files.
- 2) **Script Window**: Use the script window to run a series of commands or a whole program.
- 3) **Workspace**: The MATLAB workspace consists of the set of variables (named arrays) built up during a MATLAB session and stored in memory. You add variables to the workspace by using functions, running M-files, and loading saved workspaces.



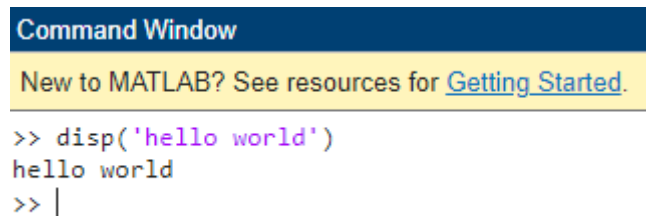
## MATLAB Basic Commands

- 1) **disp()**: To display a string

### Program:

```
disp('hello world')
```

### Output:



```
Command Window
New to MATLAB? See resources for Getting Started.
>> disp('hello world')
hello world
>> |
```

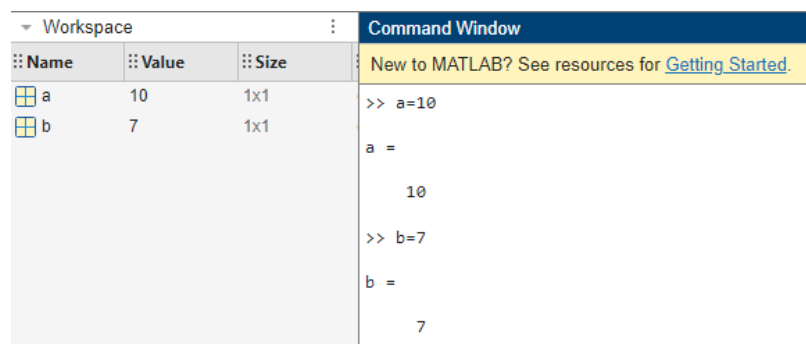
- 2) **Variables:** MATLAB does not require any type declarations or dimension statements. When MATLAB encounters a new variable name, it automatically creates the variable and allocates the appropriate amount of storage. If the variable already exists, MATLAB changes its contents and, if necessary, allocates new storage. For example, **a=10** creates a 1-by-1 matrix named a and stores the value 10 in its single element.

Variable names consist of a letter, followed by any number of letters, digits, or underscores. MATLAB uses only the first 31 characters of a variable name. MATLAB is case sensitive; it distinguishes between uppercase and lowercase letters. A and a are not the same variable. To view the matrix assigned to any variable, simply enter the variable name.

### Program:

```
a=10
b=7
```

### Output:



Workspace		
Name	Value	Size
a	10	1x1
b	7	1x1

```
Command Window
New to MATLAB? See resources for Getting Started.
>> a=10
a =
    10
>> b=7
b =
     7
```

- 3) **clc**: To clear the command window screen.
- 4) **clear**: To clear all the workspace variables from memory. We can also clear a particular variable from memory by explicitly mentioning the variable name like **clear a**.
- 5) **Display a stored variable**: Like we displayed string using disp() we can also display a variable by using the variable.

**Program:**

```
disp(a)
```

- 6) **Display a combination of variable and string:** We can also display a combination of variable and string by using concatenation [ ] and a command **num2str()** that can convert the integer to string.

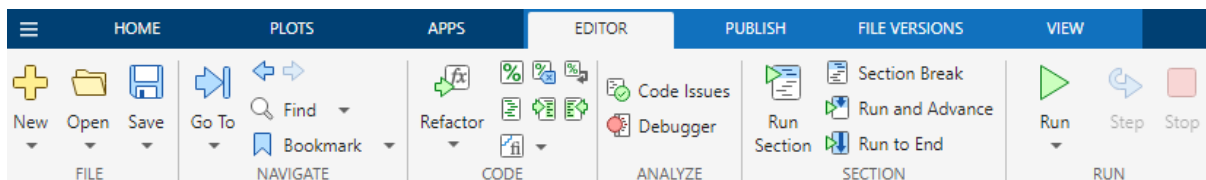
**Program:**

```
disp(['The value of a is ',num2str(a)])
```

**Output:**

```
Command Window
New to MATLAB? See resources for Getting Started.
>> disp(['The value of a is ',num2str(a)])
The value of a is 10
>> |
```

- 7) **Writing a program or a script:** We can write a series of command/ script/ program using the script window. To run such script we have to follow the below steps:



- First navigate to the editor menu.
  - Click on New.
  - Then click on script option
  - Save it as a .m file.
  - Write the script.
  - Run it from the Run command in the editor menu.
- 8) **Take input from user:** We can also take input from a user rather than assigning it. We will write a script for it.

**Program:**

```
prompt = "Enter the value of x? ";
x = input(prompt)
```

**Output:**

```
Command Window
Enter the value of x?
50

x =

    50
```








9) **Operators:** MATLAB uses familiar arithmetic operators

- +** Addition
- Subtraction
- \*** Multiplication
- /** Division
- ^** Power
- \** Complex conjugate transpose
- ()** Specify evaluation order

**Program:**

```
a=10;
b=5;
sum=a+b;
sub=a-b;
div=a/b;
pow=a^b;
exp=(a+b)*(a-b);
```

**Output:**

Workspace			
Name	Value	Size	
 a	10	1x1	
 b	5	1x1	
 div	2	1x1	
 exp	75	1x1	
 pow	100000	1x1	
 sub	5	1x1	
 sum	15	1x1	

## Matrix Operations

### Generating Matrices

MATLAB provides four functions that generate basic matrices.

- **zeros** All zeros
- **ones** All ones
- **rand** Uniformly distributed random elements
- **randi** Uniformly distributed random integers

### Program:

**zeros(5):** Generate a square matrix of size 5 with all zeros.

### Output:

```
>> zeros(5)

ans =

     0     0     0     0     0
     0     0     0     0     0
     0     0     0     0     0
     0     0     0     0     0
     0     0     0     0     0
```

**zeros(2,3):** Generate a 2\*3 matrix with all zeros.

### Output:

```
>> zeros(2,3)

ans =

     0     0     0
     0     0     0
```

**5\*ones(5,4):** Generate a 5\*4 matrix with all ones.

### Output:

```
>> 5*ones(5,4)

ans =

     5     5     5     5
     5     5     5     5
     5     5     5     5
     5     5     5     5
     5     5     5     5
```

**rand(5):** Generate a square matrix of size 5 with random numbers between 0 and 1.

**Output:**

```
>> rand(5)

ans =

    0.8010    0.5785    0.5211    0.3955    0.9133
    0.0292    0.2373    0.2316    0.3674    0.7962
    0.9289    0.4588    0.4889    0.9880    0.0987
    0.7303    0.9631    0.6241    0.0377    0.2619
    0.4886    0.5468    0.6791    0.8852    0.3354
```

**rand(1,10):** Generate a 1\*10 matrix with random numbers between 0 and 1.

**Output:**

```
>> rand(1,10)

ans =

    0.6797    0.1366    0.7212    0.1068    0.6538    0.4942    0.7791    0.7150    0.9037    0.8909
```

**randi(10,5):** Generate a 5-by-5 matrix of random integers between 1 and 10. The first input to randi indicates the largest integer in the sampling interval (the smallest integer in the interval is 1).

**Output:**

```
>> randi(10,5)

ans =

     4     6     9     9     8
     7     5     9     1     6
     2    10     6     5     5
     1     7     2     2     1
     8     7     3    10     7
```

**randi([-5,5],5,5):** Generate a 5\*5 matrix of uniformly distributed random integers from the sample interval [-5,5].

**Output:**

```
>> randi([-5,5],5,5)

ans =

    -1    -2    -5     0     2
     4    -4    -3     0     5
     2     1    -4     4    -3
     5     3    -2     0     2
     0    -1    -1     5    -2
```

**To initialize our own matrix ( Use of comma, semicolon, and colon )**

**Program:**

```
a=[1 2 3 4]
```

```
a=[1,2,3,4]
```

**Output:**

```
>> a= [1,2,3,4]

a =

     1     2     3     4|

>> a=[1 2 3 4]

a =

     1     2     3     4
```

**Program:**

```
a=[1 2 3 4; 5 6 7 8; 9 10 11 12]
```

**Output:**

```
>> a=[1 2 3 4; 5 6 7 8; 9 10 11 12]

a =

     1     2     3     4
     5     6     7     8
     9    10    11    12
```

**Program:**

```
A=[1:10]
```

**Output:**

```
>> a=[1:10]

a =

     1     2     3     4     5     6     7     8     9    10
```

**Program:** Sum of 2 matrices

```
a=[1 2 3 4; 5 6 7 8; 9 10 11 12];  
b=[1 2 3 4; 5 6 7 8; 9 10 11 12];  
c=a+b
```

**Output:**

```
c =  
  
     2     4     6     8  
    10    12    14    16  
    18    20    22    24
```

**Program:** Transpose of a matrix

```
a=[1 2 3 4; 5 6 7 8; 9 10 11 12];  
transpose = a'
```

**Output:**

```
>> transpose = a'  
  
transpose =  
  
     1     5     9  
     2     6    10  
     3     7    11  
     4     8    12
```

**Program:** Array Index

```
a=[1:10];
```

```
a(1)
```

```
a(2)
```

**Output:**

```
>> a=[1:10]  
  
a =  
  
     1     2     3     4     5     6     7     8     9    10  
  
>> a(1)  
  
ans =  
  
     1  
  
>> a(2)  
  
ans =  
  
     2
```