INTRODUCTION TO MATLAB

Date: 20/01/2020

Declaring Variables, Constants, Integers and Strings:-

To create a new variable, enter the variable name in the Command Window, followed by an equal sign (=) and the value you want to assign to the variable.
For example, if you run these statements, MATLAB adds the two variables x and A to the workspace:

```
x = 5.71;

A = [1 2 3; 4 5 6; 7 8 9];
```

For declaring Integers or strings, the following syntax can be used:

```
I = int32(25);

S = 'This is a string';
```

Matrices:-

MATLAB is an abbreviation for "matrix laboratory." While other programming languages mostly work with numbers one at a time, MATLAB® is designed to operate primarily on whole matrices and arrays.

All MATLAB variables are multidimensional arrays, no matter what type of data. A matrix is a two-dimensional array often used for linear algebra.

To create an array with four elements in a single row, separate the elements with either a comma (,) or a space.

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

This type of array is a row vector.

To create a matrix that has multiple rows, separate the rows with semicolons.

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

Another way to create a matrix is to use a function, such as ones, zeros, or rand. For example, create a 5-by-1 column vector of zeros.

```
z = zeros(5,1);
```

Arithmetic Operators:-

Arithmetic functions include operators for simple operations like addition and multiplication, as well as functions for common calculations like summation, moving sums, modulo operations, and rounding.

Addition

```
x = 5.71 + 9.81;

A = [1 2 3; 4 5 6; 7 8 9];

y = sum(A); %Adds all elements of A
```

Subtraction

```
x = 5.71 - 6.0;
```

Multiplication

```
X = 5*5;

A = rand(5,5); %Creates a matrix of dimension 5 x 5 with random
values.

B = rand(5,5);

C = A*B; %Multiplies A and B as matrices

D = A.*B; %Multiplies individual elements of A and B
```

Division

```
x = 3/2.0;

A = rand(5,5);

B = rand(5,5);

C = A/B; %Multiplies A with B^-1

D = A./B %Divides individual values of A and B
```

Power

```
x = 3^2;

A = rand(5,5);

C = A^2; %Returns A x A

D = A.^2; %Squares individual element of A
```

Modulo

```
x = mod(10,5); %Returns remainder of 10 after dividing with 5

A = rand(5,5);

P = mod(A,2) %Returns a matrix with 1 if element of A is not divisible

by 0
```

Logical Operators:-

The logical data type represents true or false states using the numbers 1 and 0, respectively. Certain MATLAB® functions and operators return logical values to indicate fulfillment of a condition. You can use those logical values to index into an array or execute conditional code.

`Short-circuit &&, ||`	Logical operations with short-circuiting
`&`	Find logical AND
`~`	Find logical NOT
`|`	Find logical OR
`xor`	Find logical exclusive-OR
`all`	Determine if all array elements are nonzero or true
`any`	Determine if any array elements are nonzero
`false`	Logical 0 (false)
`find`	Find indices and values of nonzero elements
`islogical`	Determine if input is logical array
`logical`	Convert numeric values to logicals
`true`	Logical 1 (true)

Looping, Branching and Controlling:-

Conditional statements enable you to select at run time which block of code to execute. The simplest conditional statement is an if statement. For example:

```
%if example

a = randi(100, 1);

if rem(a, 2) == 0

    disp('a is even'); %Displays if a is even.

end

%Switch Example

[dayNum, dayString] = weekday(date, 'long', 'en_US');

switch dayString

   case 'Monday'

      disp('Start of the work week')

   case 'Tuesday'

      disp('Day 2')

   case 'Wednesday'

      disp('Day 3')

   case 'Thursday'

      disp('Day 4')

   case 'Friday'

      disp('Last day of the work week')

   otherwise

      disp('Weekend!')

end
```

With loop control statements, you can repeatedly execute a block of code. There are two types of loops:

For loops

```
x = ones(1,10);

for n = 2:6

    x(n) = 2 * x(n - 1);

end
```

While loops

```
n = 1;

nFactorial = 1;

while nFactorial < 1e100

    n = n + 1;

    nFactorial = nFactorial * n; %Used to find factorial less than

1e100

end
```

Functions:-

Both scripts and functions allow you to reuse sequences of commands by storing them in program files. Scripts are the simplest type of program, since they store commands exactly as you would type them at the command line.
Functions provide more flexibility, primarily because you can pass input values and return output values. For example, this function named fact computes the factorial of a number (n) and returns the result (f).

`function [y1,...,yN] = myfun(x1,...,xM)` declares a function named `myfun` that accepts inputs `x1,...,xM` and returns outputs `y1,...,yN`.

This declaration statement must be the first executable line of the function.
Valid function names begin with an alphabetic character, and can contain letters, numbers, or underscores.

Example:-
Define a function in a file named average.m that accepts an input vector, calculates the average of the values, and returns a single result.

```
function ave = average(x)

    ave = sum(x(:))/numel(x);

end
```

Plotting Commands:-

`plot(X, Y)` creates a 2-D line plot of the data in Y versus the corresponding values in X.
If X and Y are both matrices, then they must have equal size. The plot function plots columns of Y versus columns of X.

- If one of X or Y is a vector and the other is a matrix, then the matrix must have dimensions such that one of its dimensions equals the vector length. If the number of matrix rows equals the vector length, then the plot function plots each matrix column versus the vector. If the number of matrix columns equals the vector length, then the function plots each matrix row versus the vector. If the matrix is square, then the function plots each column versus the vector.
- If one of X or Y is a scalar and the other is either a scalar or a vector, then the plot function plots discrete points. However, to see the points you must specify a marker symbol, for example, `plot(X, Y, 'o')`

Example:

```
x = 0:pi/100:2*pi;

y = sin(x);

subplot(1,1,1); %Used to display more than 1 plots in a single figure.

plot(x,y);

title('2-D Line Plot') %Used to provide title to plot

xlabel('x') %Used to provide X Axis label

ylabel('cos(5x)') %Used to provide Y Axis label
```