<u>OCTAVE COMMANDS</u>

- **~=** // Not Equal
- **%** Message // comments
- **clc** // clear terminal
- **PS1(‘>> ’)** // to change terminal input symbol
- **0** // false
- **1** // true
- **==** // comparison

Variables:

- **a = value** //prints value
- **a = value;** // with semicolon it suppresses the printing
- **b = ‘String’;** // for string assignment
- **b** // print value of b
- **a=pi** // assign value of PI into a
- **a** // print value of pi

  <u>Out</u> a = 3.1416
- **disp(a)** // print statement

  <u>Out</u> = 3.1416

- **disp(sprintf('2 decimal: %0.2f', a))**

  // displays sprint of variable

  <u>Out</u> = 2 decimal: 3.14

  // means display decimal values according to value given

- **format long**
- **a**

  // displays long value of a

  <u>Out</u> = a = 3.141592653589793

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

  // creates matrix

  // displays long value of a

  <u>Out</u> =

  A =

  1 2

  3 4

  5 6

  7 8

- **v = 1:0.1:2**

// creates matrix from 1 and do increment by 0.1 until 2

<u>Out</u> =

Columns 1 through 5:

1.000000000000000 1.100000000000000 1.200000000000000 1.300000000000000 1.400000000000000

Columns 6 through 10:

1.500000000000000 1.600000000000000 1.700000000000000 1.800000000000000 1.900000000000000

Column 11:

2.000000000000000

- **v = 1:6**

// creates matrix up to 6 starting from 1

<u>Out</u> =

v =

1 2 3 4 5 6

- **ones(2,3)**

  // creates a matrix of 1 of order 2 x3

<u>Out</u> =

1 1 1

1 1 1

1 1 1

- **rand(2,3)**

  // creates a matrix of a random number of order 2 x3

<u>Out</u> =

0.173784097592126	**0.296735944466070**	**0.792266003772464**
0.695541334906015	**0.888444304837793**	**0.829518710553532**

- **randn(2,3)**

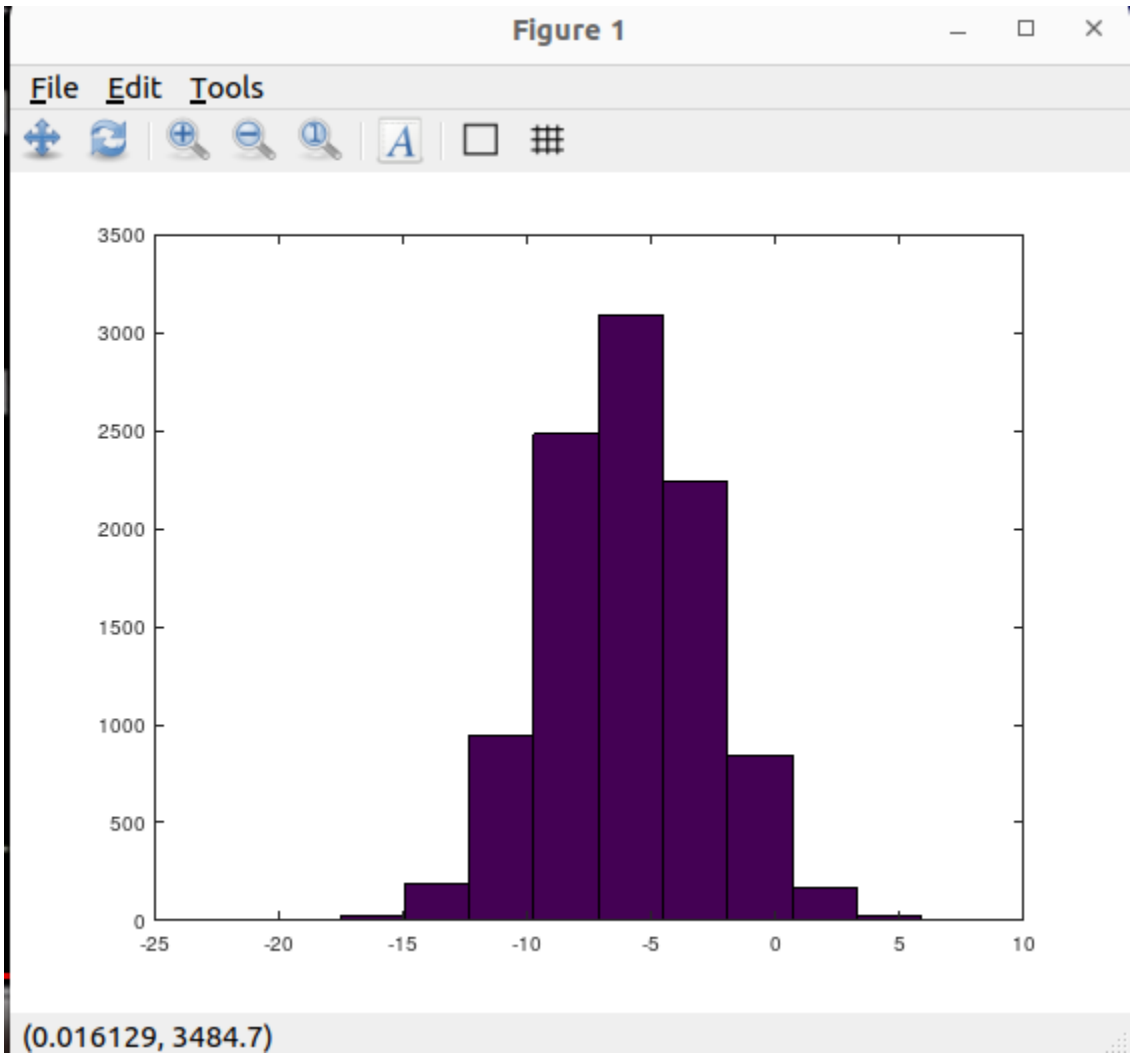
  // creates a matrix of a random number of order 2 x3 of Gaussian Random Values

<u>Out</u> =

-3.608487993740562e-01	**2.911926384437561e+00**	**2.213822526036592e+00**
4.143580052077651e-01	**-1.717117983840558e+00**	**8.770997108535140e-02**

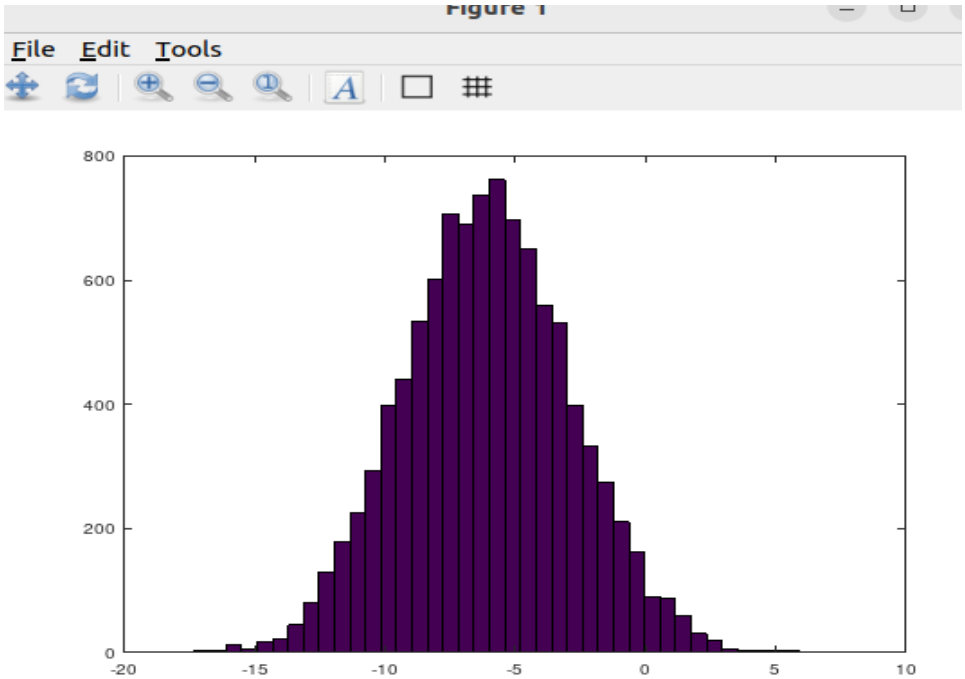
- **w = -6 + sqrt(10)&(randn(1,1000));**
- **hist(w)**

  //creates a histogram of values from 1 to 1000 with mean -6



- **hist(w, 40)**

  //creates a histogram of more buckets as per parameter



- **eye(4)**

  //creates a 4x4 identity matrix

  <u>Out</u> =

  Diagonal Matrix

1	**0**	**0**	**0**
0	**1**	**0**	**0**
0	**0**	**1**	**0**
0	**0**	**0**	**1**

- **size(A)** //tells the size of the matrix
- **sz = size(A)** //create a matrix of elements from size

  <u>Out</u> =

  1 2
- **size([matrix variable],1)**

  // gives the dimension of rows
- **size([matrix variable],2)**

  // gives the dimension of columns
- **length([vector variable])**

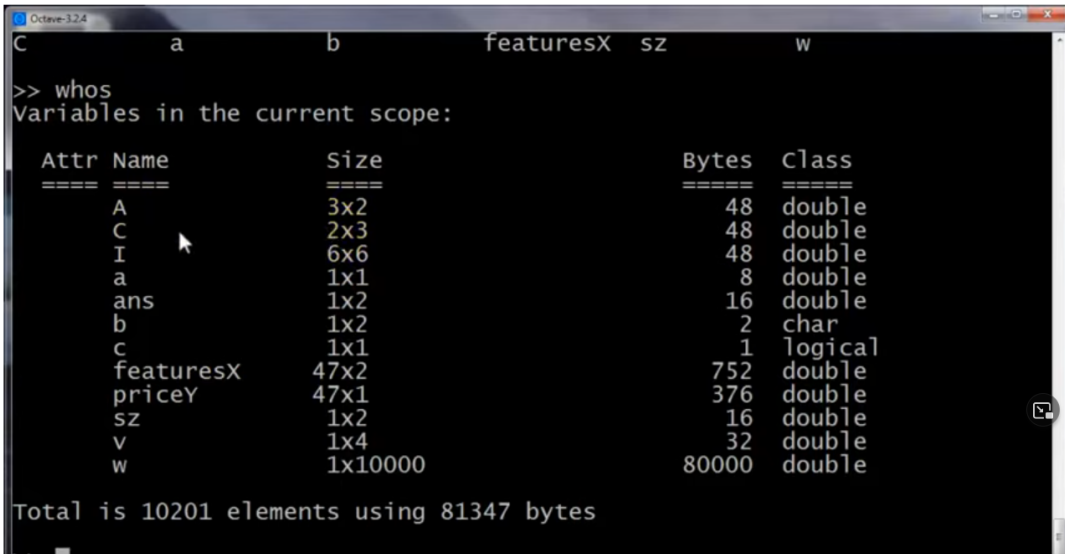
  // gives the length of vector

  // we usually apply this with vectors
- **who** //tells the variables that we have in our scope of octave space
- **load filename.ext**

  //load file present in the local working directory
- **filename**

  // display file content
- **whos**

  // gives the detail view



- **clear variableName**

  // gets rid of the variable from the scope

- **variable = filename/list_name/(1:10)**

  // assigns data of assigned variable into new variable according to given range

- **save filename.ext above_variable_name**

  // saves that as a file in the current directory

- **clear**

  // clears all variables present in scope

- **save filename.ext variabel_name ascii** //format

  // saves that file in the given format

- **matrix_name(row_num,column_num)**

  // gives the element present in that index
- **matrix_name(row_num,:)**

  // gives all the element present in that row
- **matrix_name(:,col_num)**

  // gives all the element present in that column
- **matrix_name([row_num1,row_num2] , :)**

  // gives all the elements in mentioned rows
- **matrix_name(: , [col_num1,col_num2])**

  // gives all the elements in mentioned col
- **matrix_name(: , [col_num1,col_num2])**

  // gives all the elements in ment
- **matrix_name(:,col_num) = [15; 42; 24; 65]**

  // assigns the given elements in the mentioned column
- **matrix_name(row_num,:) = [15; 24]**

  // assigns the given elements in the mentioned row
- **matrix_name = [matrix_name, [55; 849; 926; 22]];**

  // appends new col to the matrix

- **matirx_name = [matirx, another_matrix];**

  // appends new col to the matrix of another matirx

  Example >> A = [A,[55; 22; 55; 89;]];

- **matrix_name(:)**

  // puts all elements into the vector

- **C = [matirx_A matix_B]**

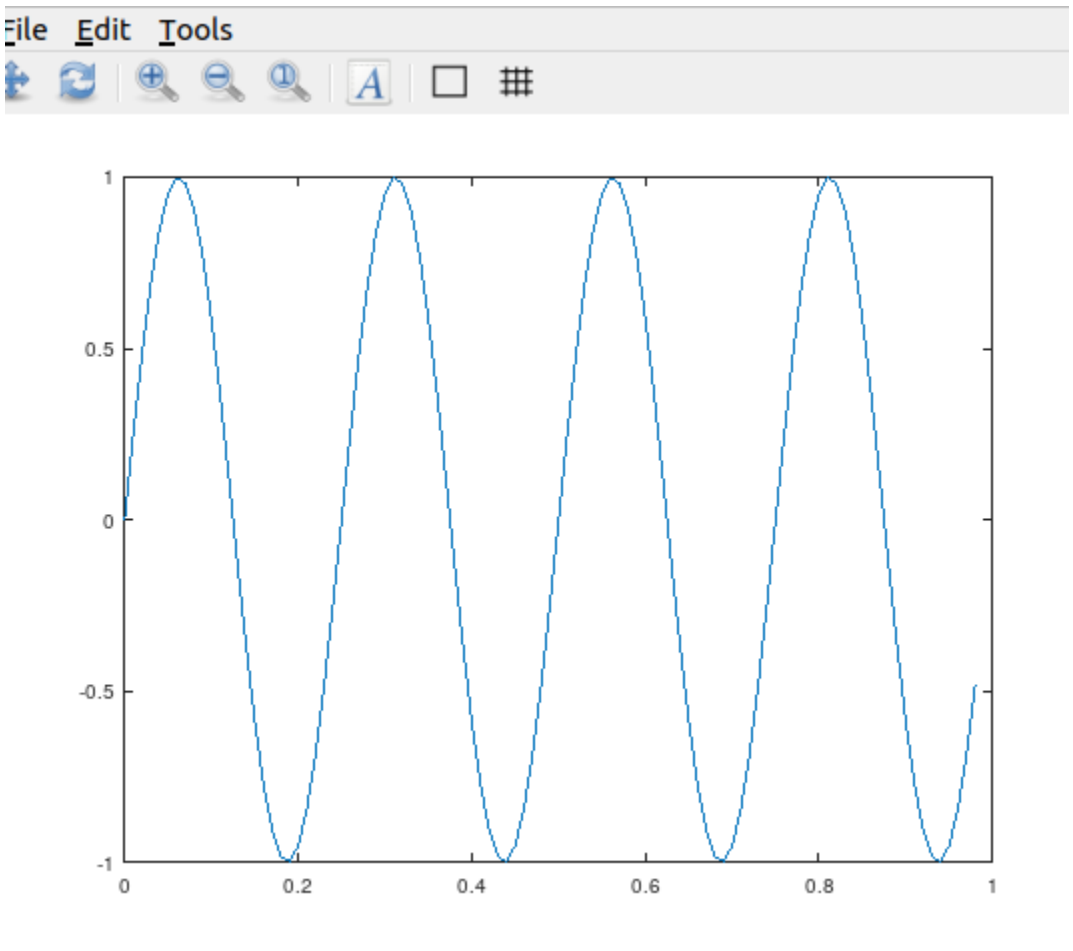
  // concatenating matrix into another matrix

- **C = [matirx_A; matix_B]**

  // puts another matrix at the end

- **variableName = [0:0.01:0.98];** //range
- **varibleName = sin(2*pi*4*t);** //Sin function
- **plot(firstVarible, secondVariable);**

<u>Out</u> =



- **hold on;**

  //hold to the plotted ui window

- **xlabel('time')**
- **ylabel('values')**

  // used for label the axis

- **legend(‘functionA’, ‘functionB’)**

  // used for setting the window of functions UI
- **title(name)**

  // used for label giving title
- **print -dpng test.png**

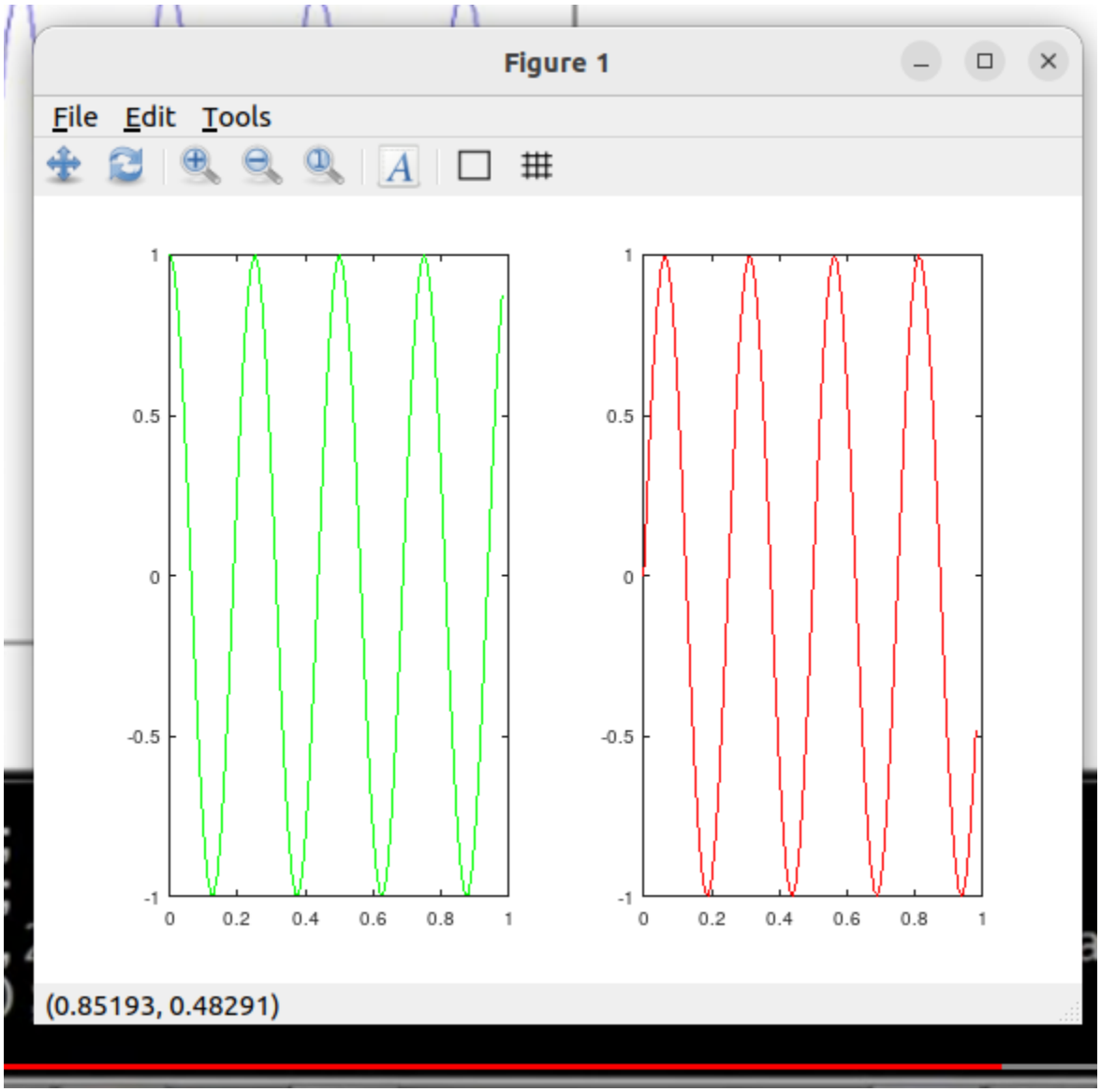
  // create png file in working directory
- **figure(1); plot(variable,variable);**

  // plots figurewise so that diff. Figures can be plotted individually
- **subplot(1,2,1);**

  // divides plot a 1x2 grid, access first element

Example :=

```
::>> subplot(1,2,1);
::>> plot(bat,amd,'g');
::>> subplot(1,2,2);
::>> plot(bat,st,'r');
```



- **axis([0.5 1 -1 4])**

  // sets range according to given number of x & y axis

- **clf**

  // clears the figure

- **matrix_name = magic(5)**

  // clears the fig

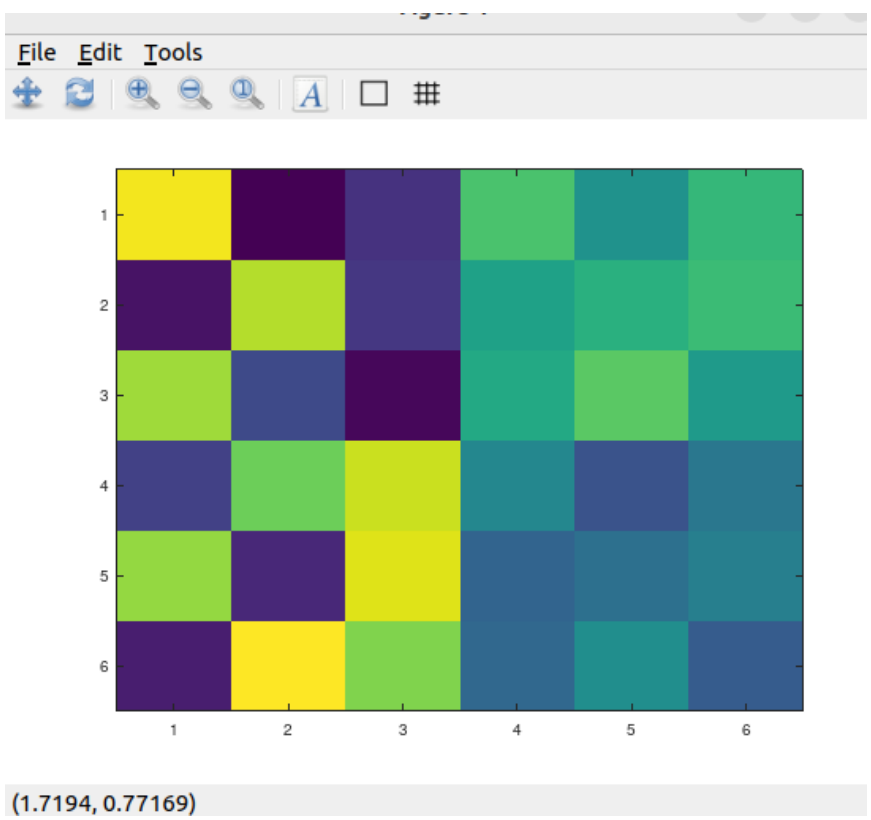
- **matrix_name = magic(6)**

  // creates random matrix of 6 x 6 order

- **imagesc(matirx_name)**

  // create grid of color where diff. Colors correspond to diff. values

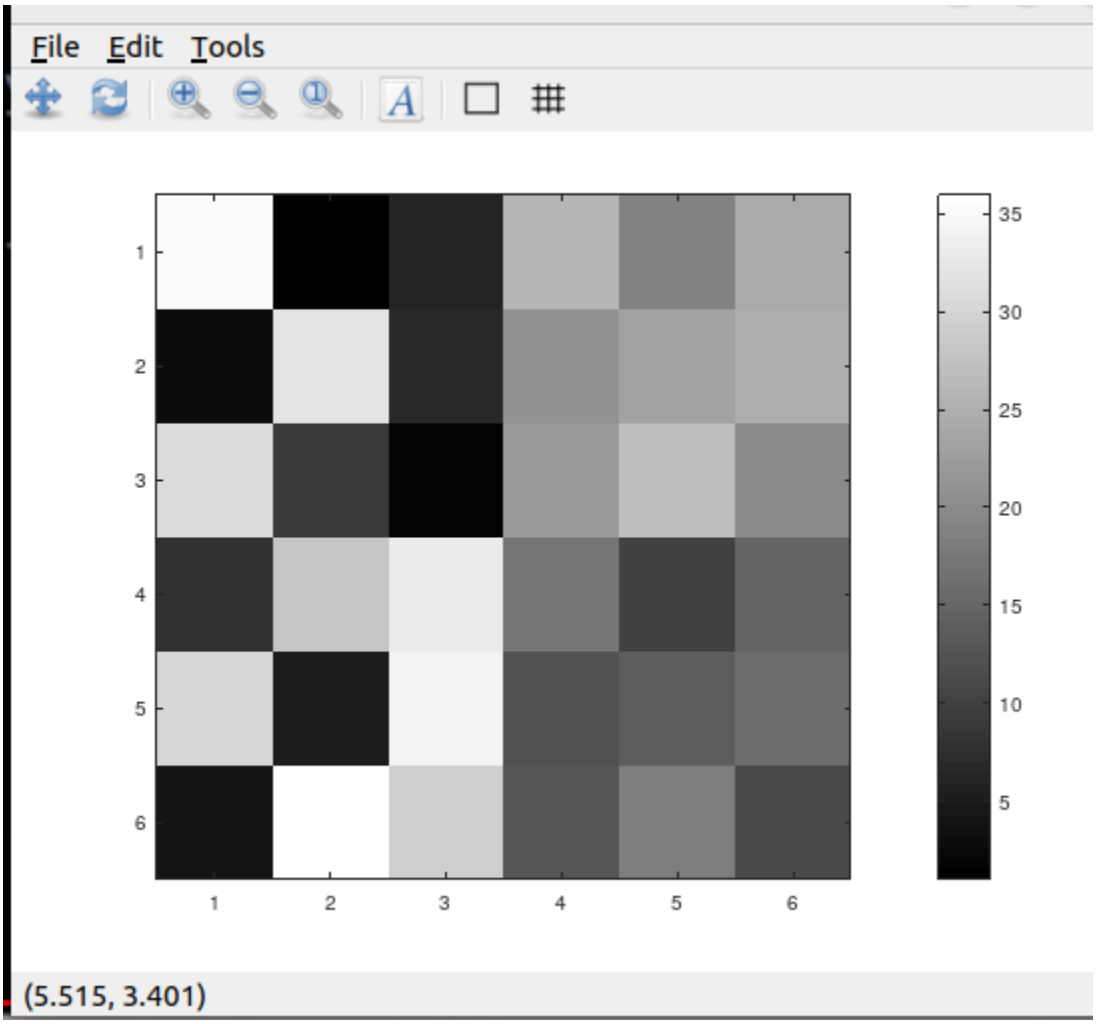
<u>Out</u> =



- **imagesc(matirx_name), colorbar, colormap color_name**

  // also create color bar and sets color as per mentioned in color name.

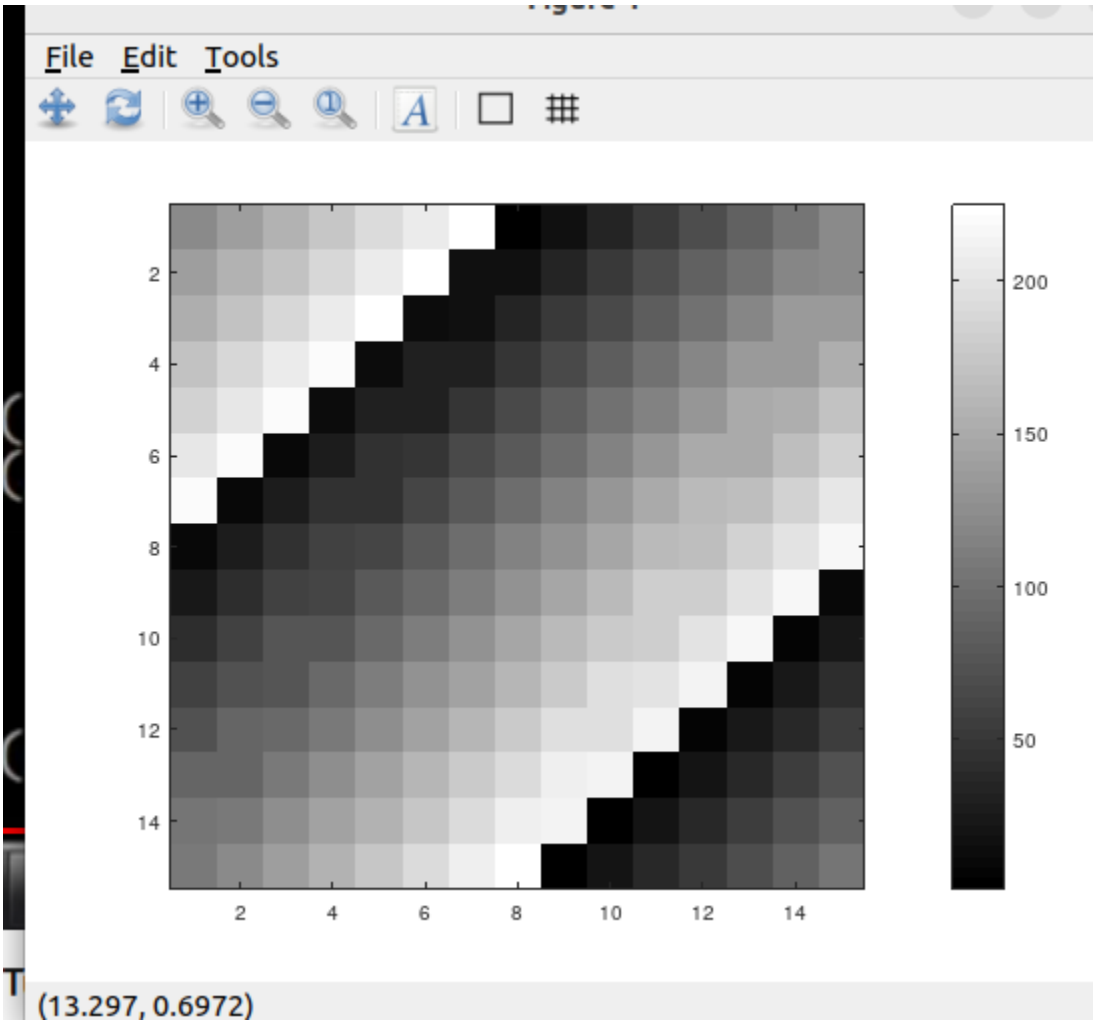
  <u>Out</u> =



- **imagesc(magic(15)), colorbar, colormap color_name;**

  // it is for representing the large order matrix view

<u>Out</u> =



Loops

- **matrix = ones(row,col)**

```
::>> for i = 1:15,          // for loop start
>  matrix(i) = 2^i;        // means 2 to the power of i
> end;                      // ends loop
```

- **indices = 1:10;**

```
::>> for i = indices,          // for loop start
>  disp(i)      // shows  elements of matrix
> end;                      // ends loop
```

- **i = 1;**

```
::>> while i <= 10,
> aman(i) = 100;
> i = i+1;
> end;
```

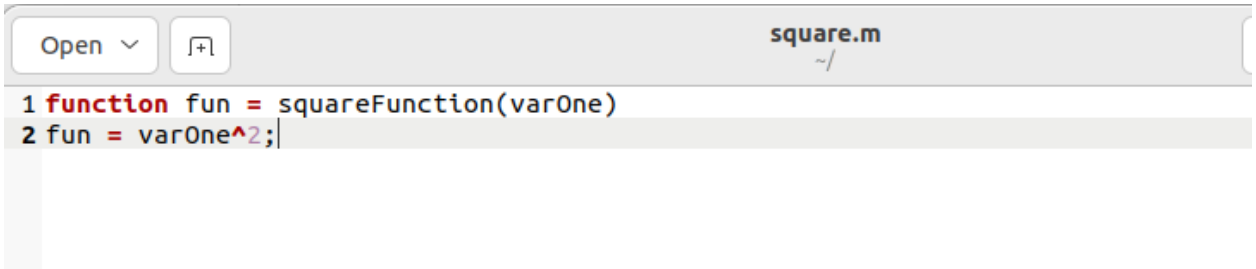
```
::>> while i <= 15    //  while loop with if
> aman(i) = 2+pi;
> i += 1;
>     if i == 8,
>             break;
>      end;
> end;
```

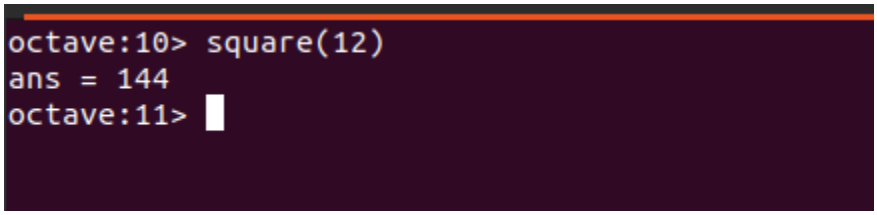
- **// If else Loop**

```
>> v(1)
ans =  999
>> v(1) = 2;
>> if v(1)==1,
>    disp('The value is one');
>  elseif v(1) == 2,
>    disp('The value is two');
>  else
>    disp('The value is not one or two.');
>  end;
```

Functions

For using function in Ocatave create a file of function in you working directory and use them in octave.





- **addpath("path")**

  // use this to add path to directory to use your file or functions located there.

- **Vectorization:**

  Vectorization is the process of converting textual data into numerical vectors and is a process that is usually applied once the text is cleaned. It can help improve the execution speed and reduce the training time of your code.

Vectorization techniques

There are three major methods for performing vectorization on text data:

1. CountVectorizer
2. TF-IDF (Term Frequency & Inverse Document Frequency)
3. Word2Vec