PRACTICAL – 01

AIM : To study and implement basic commands of MATLAB required for digital image processing techniques, and various image file formats.

Objective: The objective of the experiment is to,

- Get acquainted with Image processing commands available in the Matlab.
- How to use a given command with the help of Matlab Product Help.

Syntax: Syntax to be covered in this experiment,

1. CLC

Clear Command Window

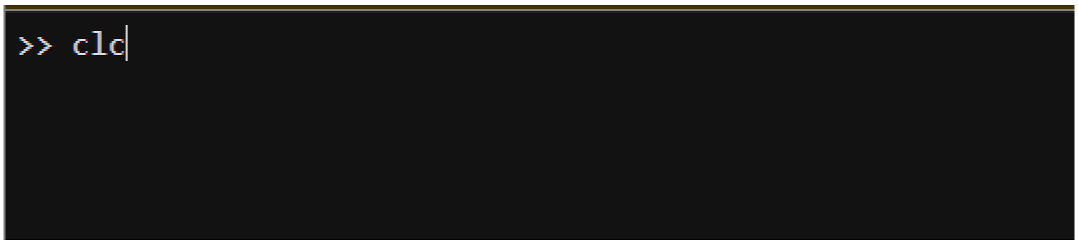
Syntax

Clc

➔ **Description**

clc clears all input and output from the Command Window display, giving you a "clean screen." After using clc, you cannot use the scroll bar to see the history of functions, but you still can use the uparrow to recall statements from the command history.

Out put:



```
>> clc
```

2. clear

- **Purpose**: Removes variables from the workspace.
- **Syntax**:

  clear

  clear name

  clear name1 name2 ...

  clear global name

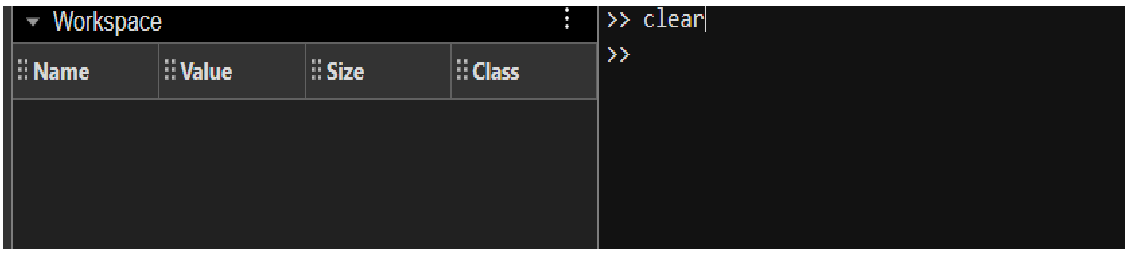
  clear -regexp expr1 expr2 ...

clear global -regexp expr1 expr2 ...

clear keyword

clear('name1', 'name2', ...)

Out put:



```
>> clear
>>
```

➔ **Description**: clear removes all variables from the workspace, freeing up system memory. Specific variables can be removed using their names, and global variables can be cleared using clear globa name.

3. close:

- **Syntax**:

close

close(h)

close name

close all

close all hidden

close all force

status = close(...)

➔ **Description**: close deletes the current or specified figure(s). close all deletes all figures, and close all hidden deletes all figures including those with hidden handles.

Out put:

```
>> close
>> |
```

4. imread

Purpose: Reads an image from a file.

Syntax :

A = imread(filename, fmt)

[X, map] = imread(...)

[...] = imread(filename)

[...] = imread(URL,...)

[...] = imread(..., Param1, Val1, ...)

Description: imread reads a grayscale or color image from a file specified by filename. The format fmt can be specified, or inferred from the file's content.

Out put:

5. Title

- **Purpose**: Adds a title to the current axes.
- **Syntax**

  title('string')

  title(fname)

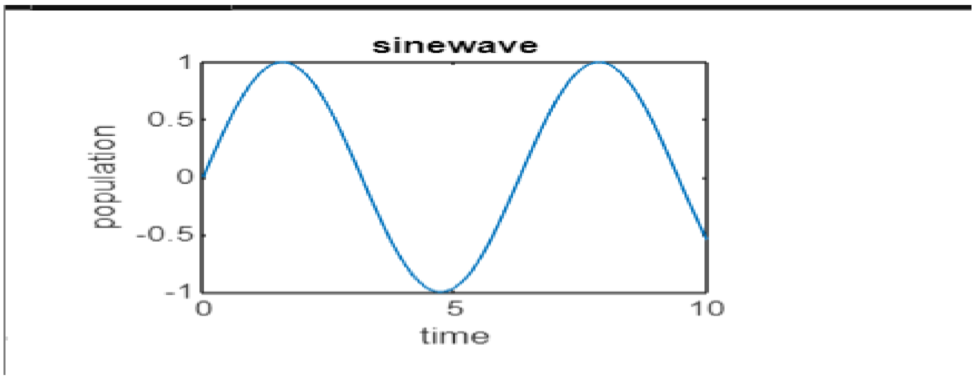
  title(..., 'PropertyName', PropertyValue, ...)

  title(axes_handle, ...)

  h = title(...)

➔ **Description**: title adds a string as a title at the top and center of the current axes. Properties for the title text can be specified.

Out put:



6. subplot

- **Purpose**: Creates axes in tiled positions.

- **Syntax**

  h = subplot(m, n, p)

  subplot(mnp)

  subplot(m, n, p, 'replace')

  subplot(m, n, P)

  subplot(h)

  subplot('Position', [left bottom width height])

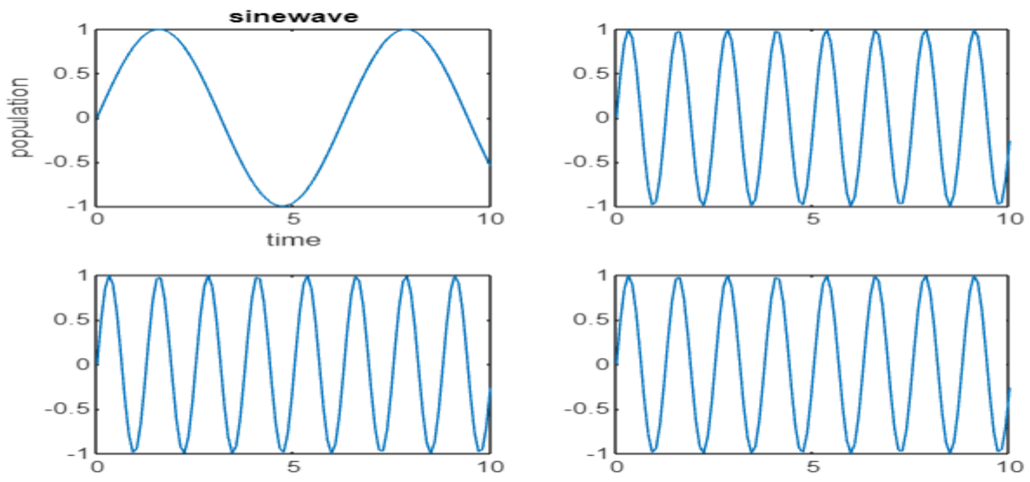
  subplot(..., prop1, value1, ...)

  h = subplot(...)

  subplot(m, n, p, 'v6')

➔ **Description**: subplot divides the figure into an m-by-n grid of axes and selects the pth axes for the current plot. It can also return the handle to the new axes object

Out put:



7. figure

Purpose: Creates a new figure window.

Syntax:
figure

figure('PropertyName', propertyvalue, ...)

figure(h)

h = figure(...)

➔ **Description**: figure creates a new figure window for graphical output. Properties of the figure can be specified, and the handle to the figure can be returned.

8. imshow

- **Purpose**: Displays an image in a figure window.
- **Syntax**

imshow(I)

imshow(I, [low high])

imshow(RGB)

imshow(BW)

imshow(X, map)

imshow(filename)

himage = imshow(...)

imshow(..., param1, val1, ...)

Out put:

➔ **Description**: imshow displays a grayscale, truecolor, or binary image. The display range for grayscale images can be specified, and the handle to the image object can be returned.

9) size

- **Purpose:** Returns the size of an array.
- **Syntax**

  d = size(X)

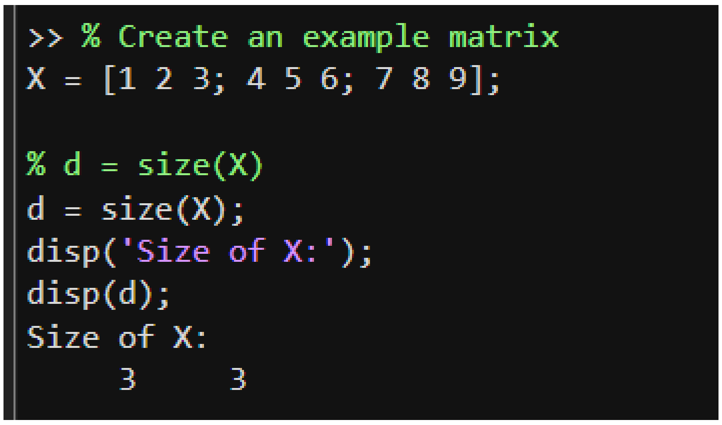
  [m, n] = size(X)

  m = size(X, dim)

  [d1, d2, d3, ..., dn] = size(X)

➔ **Description**: `size` returns the dimensions of the array `X`. It can return the size of a specific dimension or multiple dimensions.

Out put:



```
>> % Create an example matrix
X = [1 2 3; 4 5 6; 7 8 9];

% d = size(X)
d = size(X);
disp('Size of X:');
disp(d);
Size of X:
     3     3
```

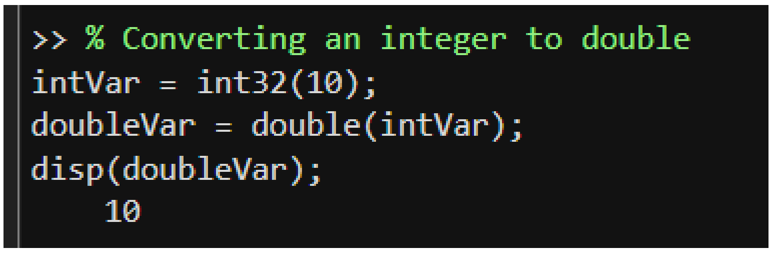
10). **double**

- **Purpose**: Converts to double precision.
- **Syntax**:

  double(x)

➔ **Description**: `double` converts the input `x` to double precision. If `x` is already double precision, there is no effect.

Out put:



```
>> % Converting an integer to double
intVar = int32(10);
doubleVar = double(intVar);
disp(doubleVar);
    10
```

11). **imagesc**

- **Purpose**: Scales data and displays it as an image.
- **Syntax**

imagesc(C)

imagesc(x, y, C)

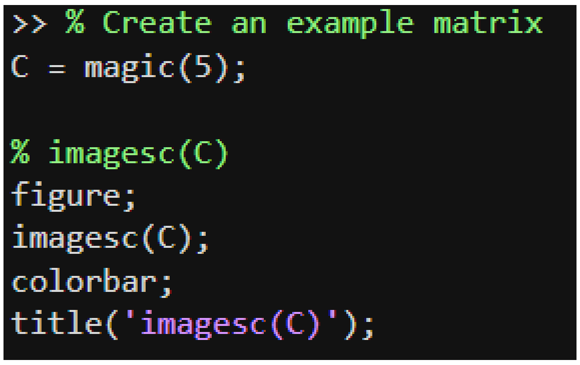
imagesc(..., clims)

imagesc('PropertyName', PropertyValue, ...)

h = imagesc(...)

➔ **Description**: `imagesc` scales the data `C` to the full range of the colormap and displays it as an image. The x and y bounds can be specified, and the handle to the image object can be returned.

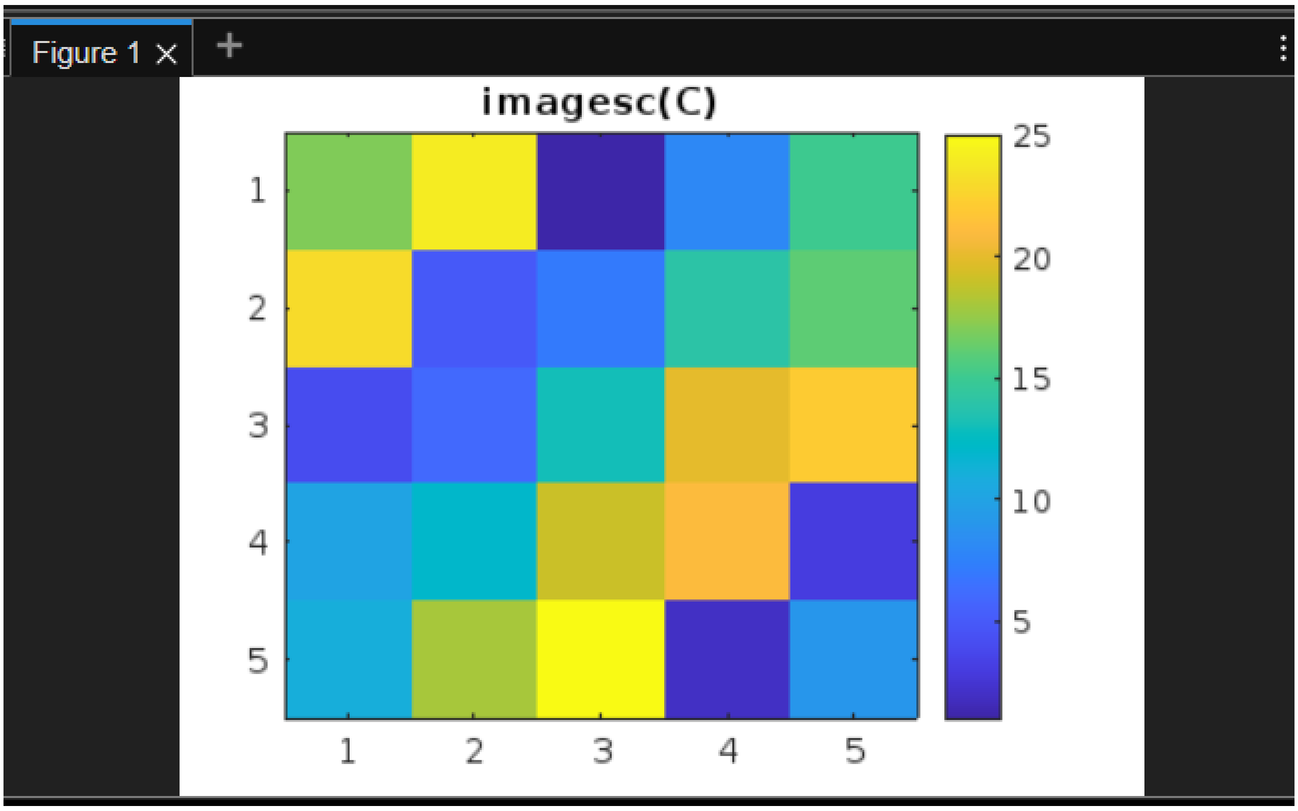
Code:



```
>> % Create an example matrix
C = magic(5);

% imagesc(C)
figure;
imagesc(C);
colorbar;
title('imagesc(C)');
```

Out put:



12). **addpath**

- **Purpose**: Adds folders to the search path.
- **Syntax**

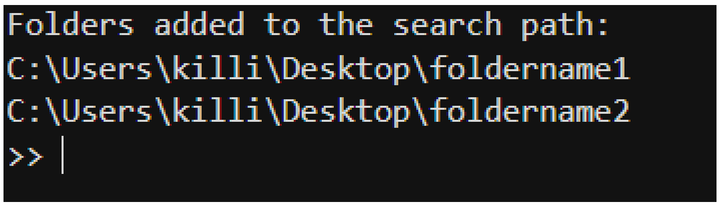
addpath('folderName1', 'folderName2', ...)

addpath('folderName1', 'folderName2', ... position)

addpath folderName1 folderName2 ... -position

➔ **Description**: `addpath` adds the specified folders to the top or bottom of the search path. It can be used in a startup file to modify the search path programmatically.

Out put:



```
Folders added to the search path:
C:\Users\killi\Desktop\foldername1
C:\Users\killi\Desktop\foldername2
>> |
```

13). **clf**

- **Purpose: Clears the current figure window.**
- **Syntax**

clf('reset')

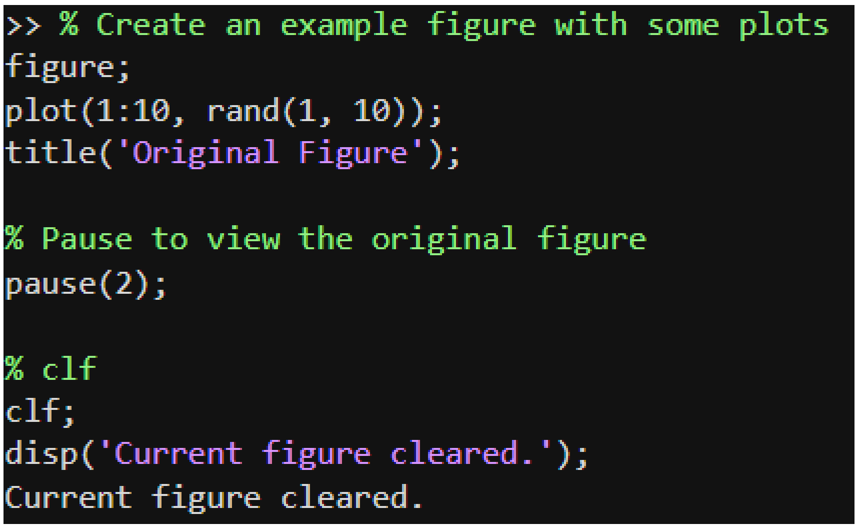
clf(fig)

clf(fig, 'reset')

figure_handle = clf(...)

➔ **Description**: `clf` deletes all graphics objects in the current figure. The `reset` option also resets all figure properties to their default values, except for certain properties

Out put:



```
>> % Create an example figure with some plots
figure;
plot(1:10, rand(1, 10));
title('Original Figure');

% Pause to view the original figure
pause(2);

% clf
clf;
disp('Current figure cleared.');
Current figure cleared.
```

14). **imwrite**

- **Purpose**: Writes an image to a file.

- **Syntax**:

  imwrite(A, filename, fmt)

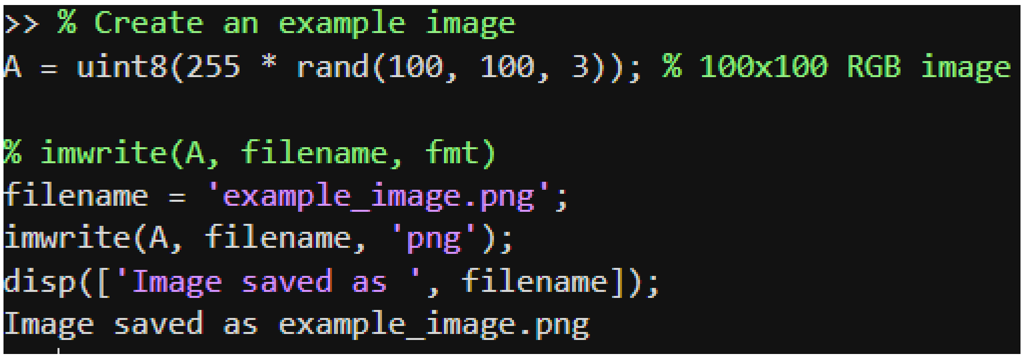
  imwrite(X, map, filename, fmt)

  imwrite(..., filename)

  imwrite(..., Param1, Val1, ...)

➔ **Description**: `imwrite` writes the image `A` to the file specified by `filename` in the format `fmt`. Parameters controlling various characteristics of the output file can be specified.

Out put:



```
>> % Create an example image
A = uint8(255 * rand(100, 100, 3)); % 100x100 RGB image

% imwrite(A, filename, fmt)
filename = 'example_image.png';
imwrite(A, filename, 'png');
disp(['Image saved as ', filename]);
Image saved as example_image.png
```

15). **imfinfo**

- **Syntax**

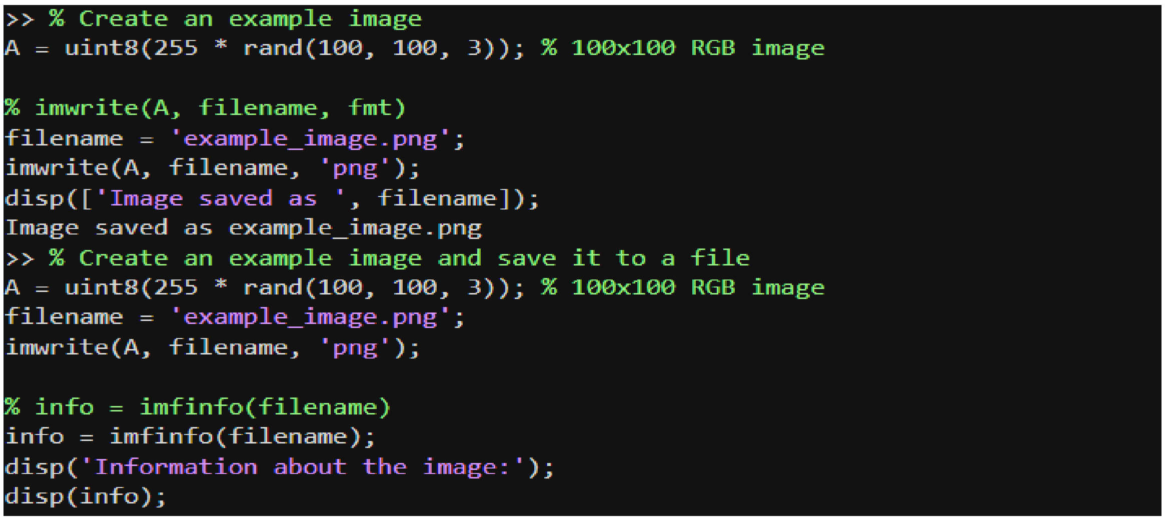
  info = imfinfo(filename, fmt)

  info = imfinfo(filename)

  info = imfinfo(URL, ...)

➔ **Description**: `imfinfo` returns a structure with information about an image file. The format `fmt` can be specified or inferred from the file content.

Code:

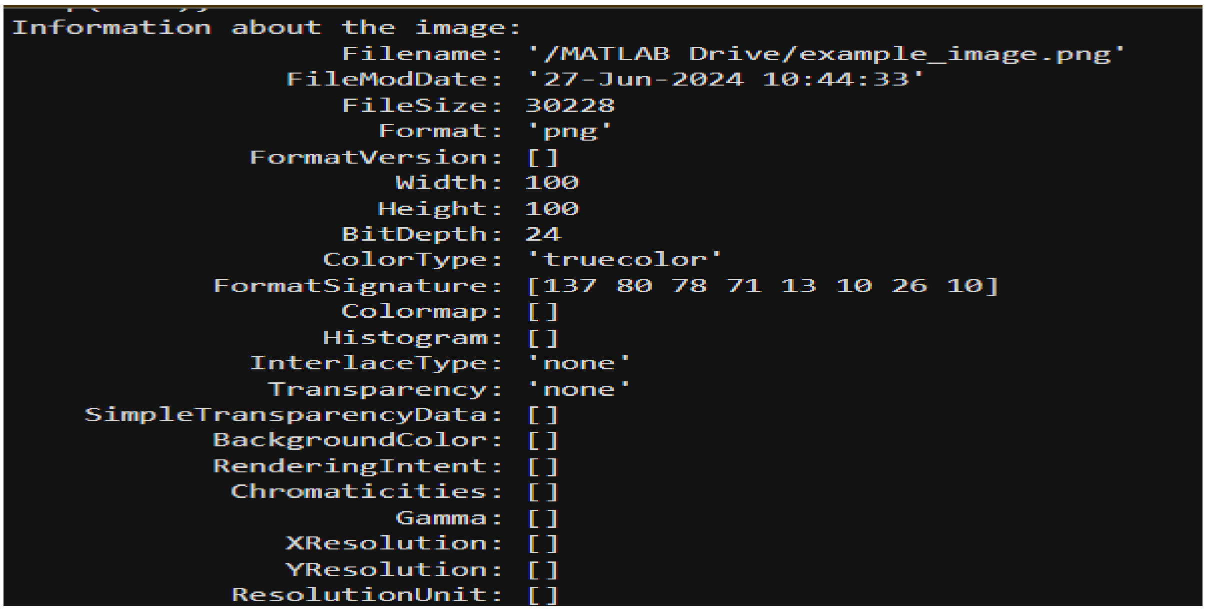


```
>> % Create an example image
A = uint8(255 * rand(100, 100, 3)); % 100x100 RGB image

% imwrite(A, filename, fmt)
filename = 'example_image.png';
imwrite(A, filename, 'png');
disp(['Image saved as ', filename]);
Image saved as example_image.png
>> % Create an example image and save it to a file
A = uint8(255 * rand(100, 100, 3)); % 100x100 RGB image
filename = 'example_image.png';
imwrite(A, filename, 'png');

% info = imfinfo(filename)
info = imfinfo(filename);
disp('Information about the image:');
disp(info);
```

Out put:



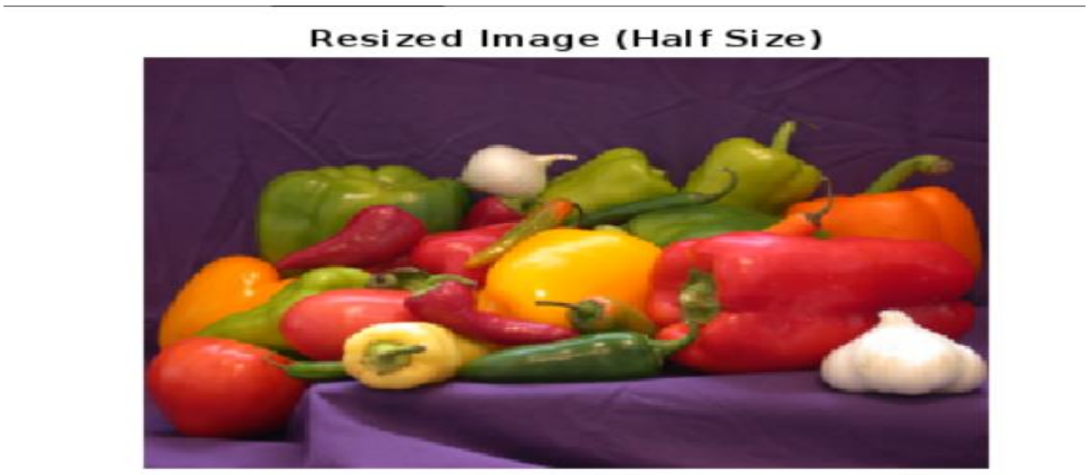
```
Information about the image:
                  Filename: '/MATLAB Drive/example_image.png'
               FileModDate: '27-Jun-2024 10:44:33'
                  FileSize: 30228
                    Format: 'png'
             FormatVersion: []
                     Width: 100
                    Height: 100
                  BitDepth: 24
                 ColorType: 'truecolor'
           FormatSignature: [137 80 78 71 13 10 26 10]
                  Colormap: []
                 Histogram: []
             InterlaceType: 'none'
              Transparency: 'none'
    SimpleTransparencyData: []
           BackgroundColor: []
            RenderingIntent: []
           Chromaticities: []
                     Gamma: []
               XResolution: []
               YResolution: []
            ResolutionUnit: []
```

16). Resize

-> Description:

To change the size of an image or video frame for various applications such as image processing, computer vision, and pre-processing for machine learning.

Out put:



Conclusion : In this practical, we explored fundamental commands in MATLAB for digital image processing. These commands allow us to manipulate, analyze, and enhance images effectively.