Computer programming and applications MATLAB-beginner

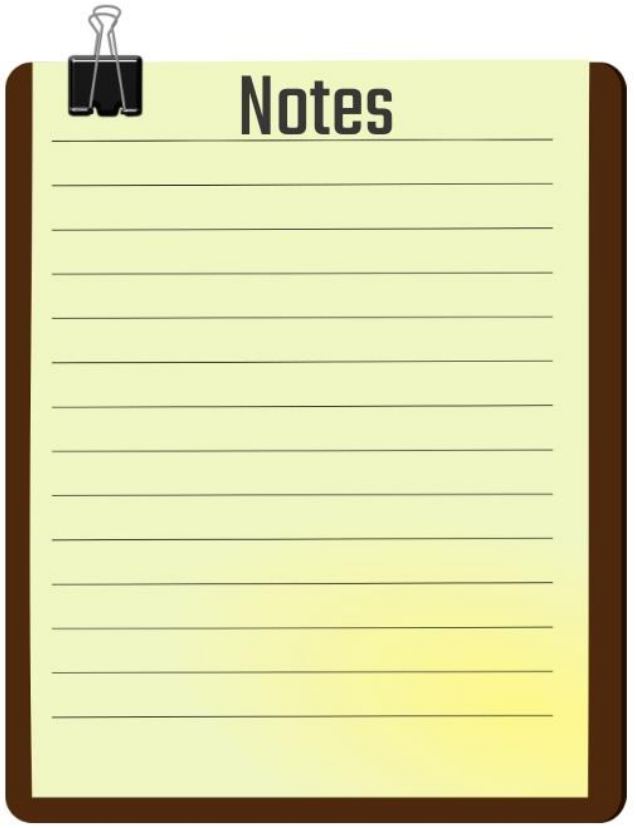
Introduction

Assist.Lec Aya Abdel Hussein

MATLAB

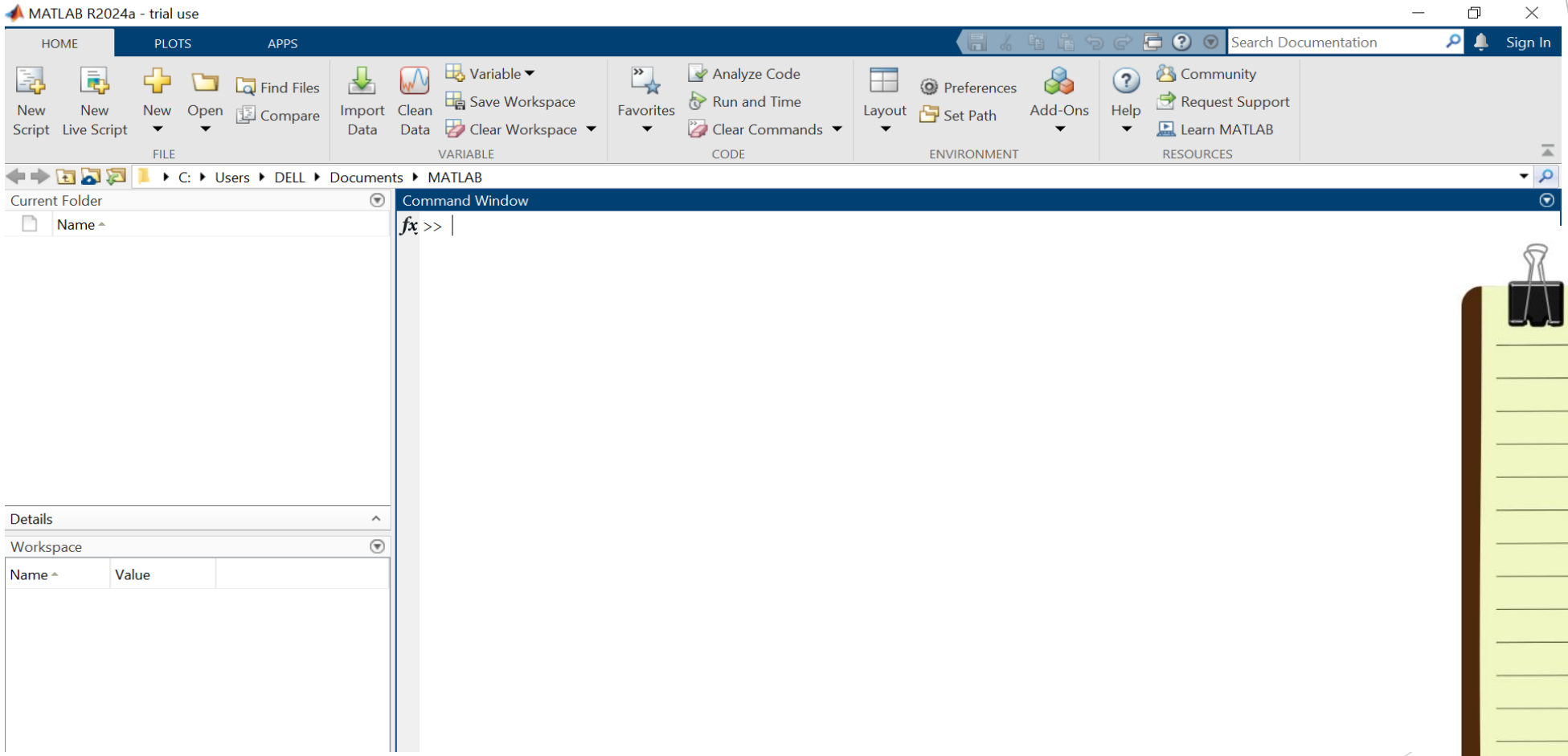
- MATLAB is a powerful computing system for handling the calculations involved in scientific and engineering problems. The name MATLAB stands for MATrix LABoratory, because the system was designed to make matrix computations particularly easy.
- One of the many things you will like about MATLAB (and which

distinguishes it from many other computer programming systems, such as C++ and Java) is that you can use it interactively. This means you type some commands at the special MATLAB prompt, and get the answers immediately. The problems solved in this way can be very simple, like finding a square root, or they can be much more complicated, like finding the solution to a system of differential equations.



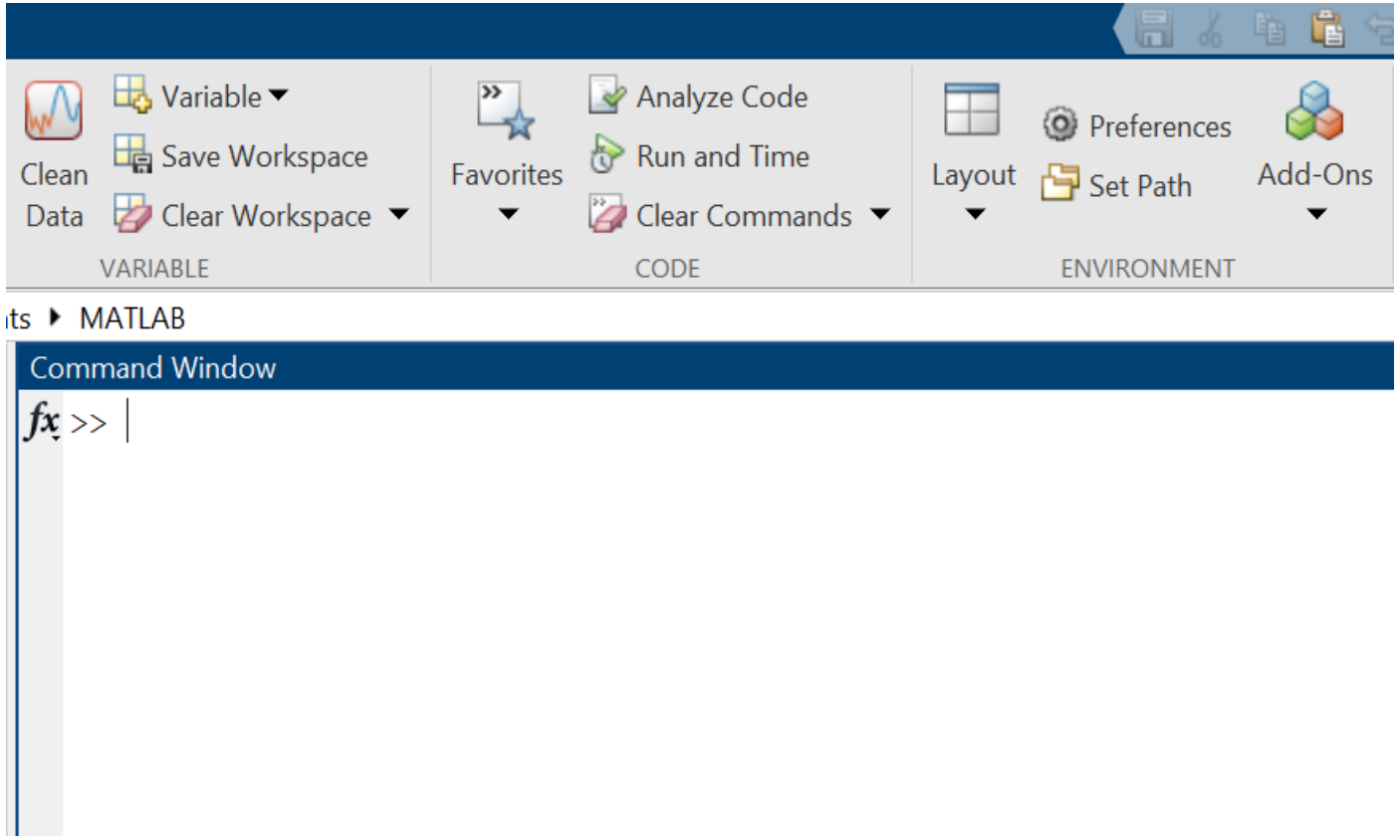
MATLAB Environments

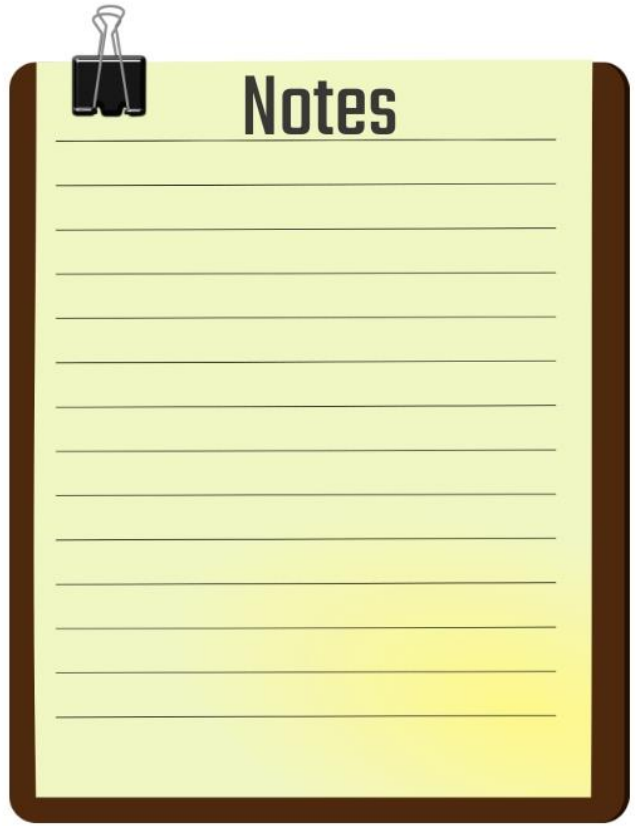
- The MATLAB Desktop is what results when you invoke MATLAB on your computer.



Matlab windows

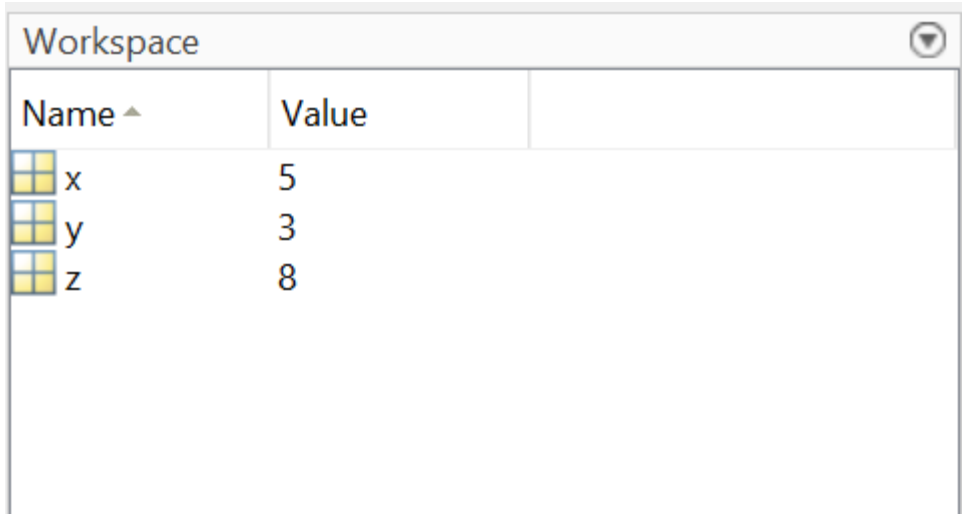
- *1. Command Window:* This window serves as the primary interface for communicating directly with MATLAB. Commands and expressions are entered here, and MATLAB evaluates them in real-time, displaying the results of computations. It's essentially the space where MATLAB and users interact.

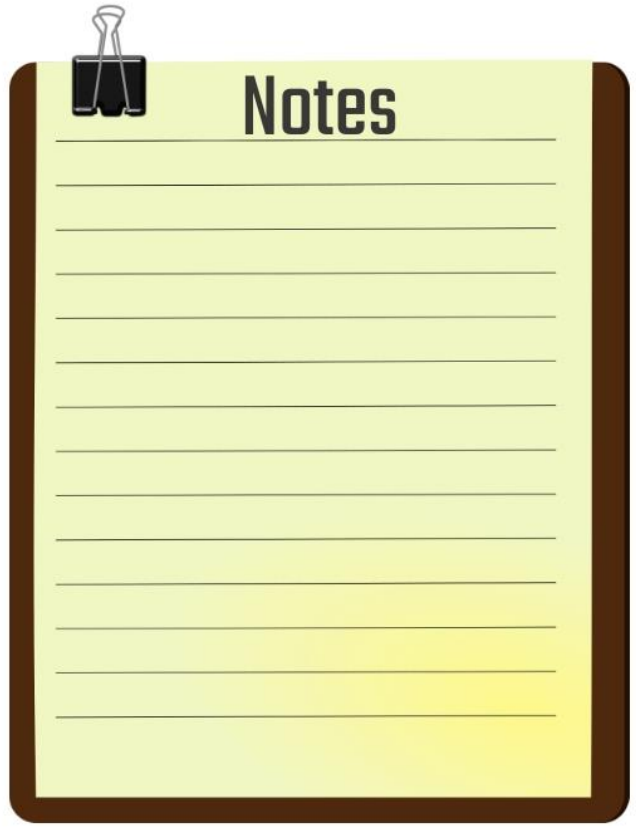




Matlab windows

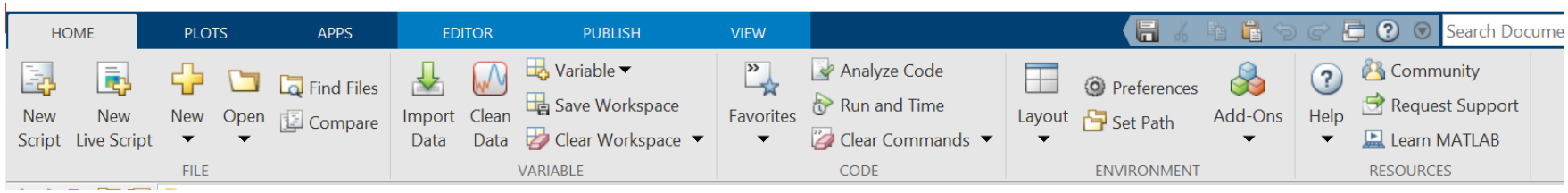
- *2. Workspace Window:* In this window, MATLAB keeps track of all variables and data used during a session. It displays various data types, with matrices being the most common. It's like a virtual table where MATLAB organizes and presents the information you're working with.

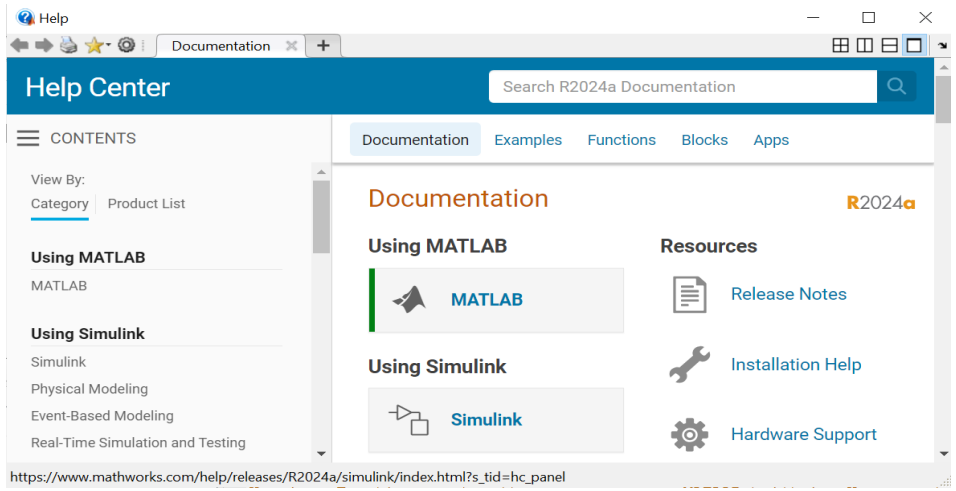


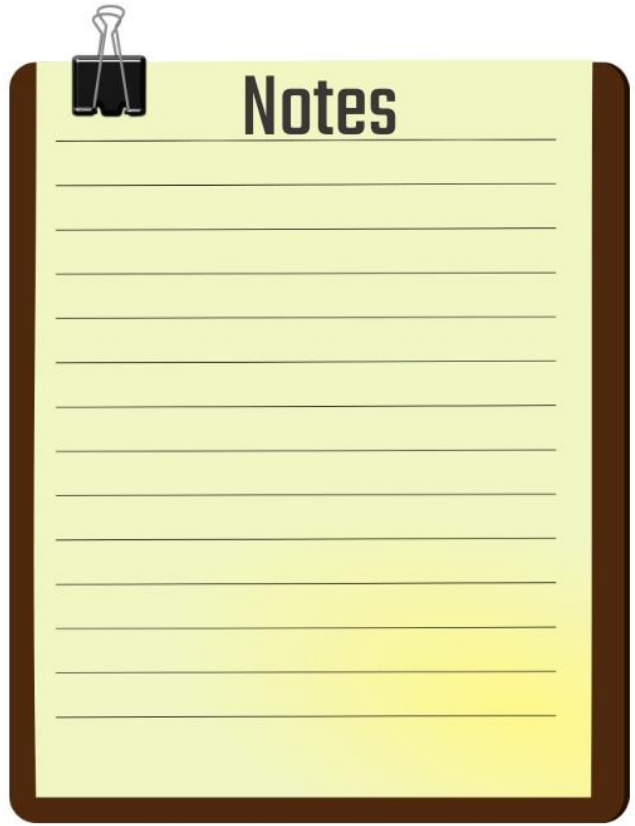


Matlab windows

- *3. Help Window:* The Help Window provides extensive information about MATLAB's language and environment, along with examples and tutorials. Users can access explanations of commands or functions, making it akin to having a comprehensive guide or tutor within MATLAB itself.

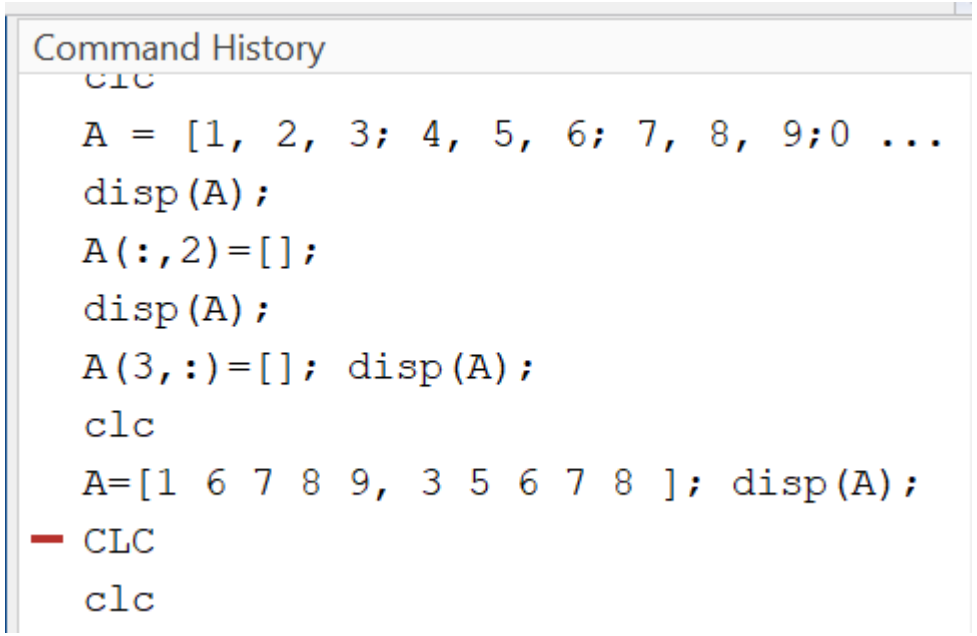


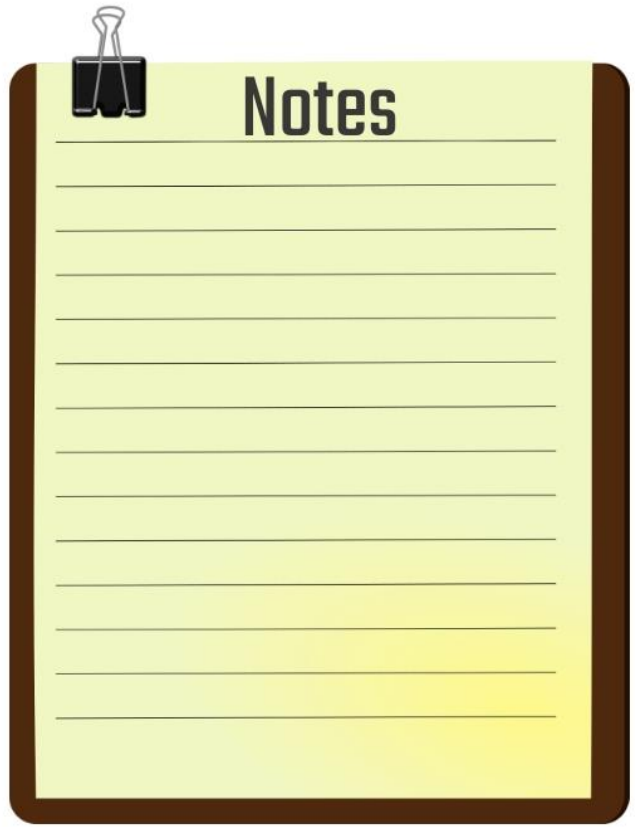




Matlab windows

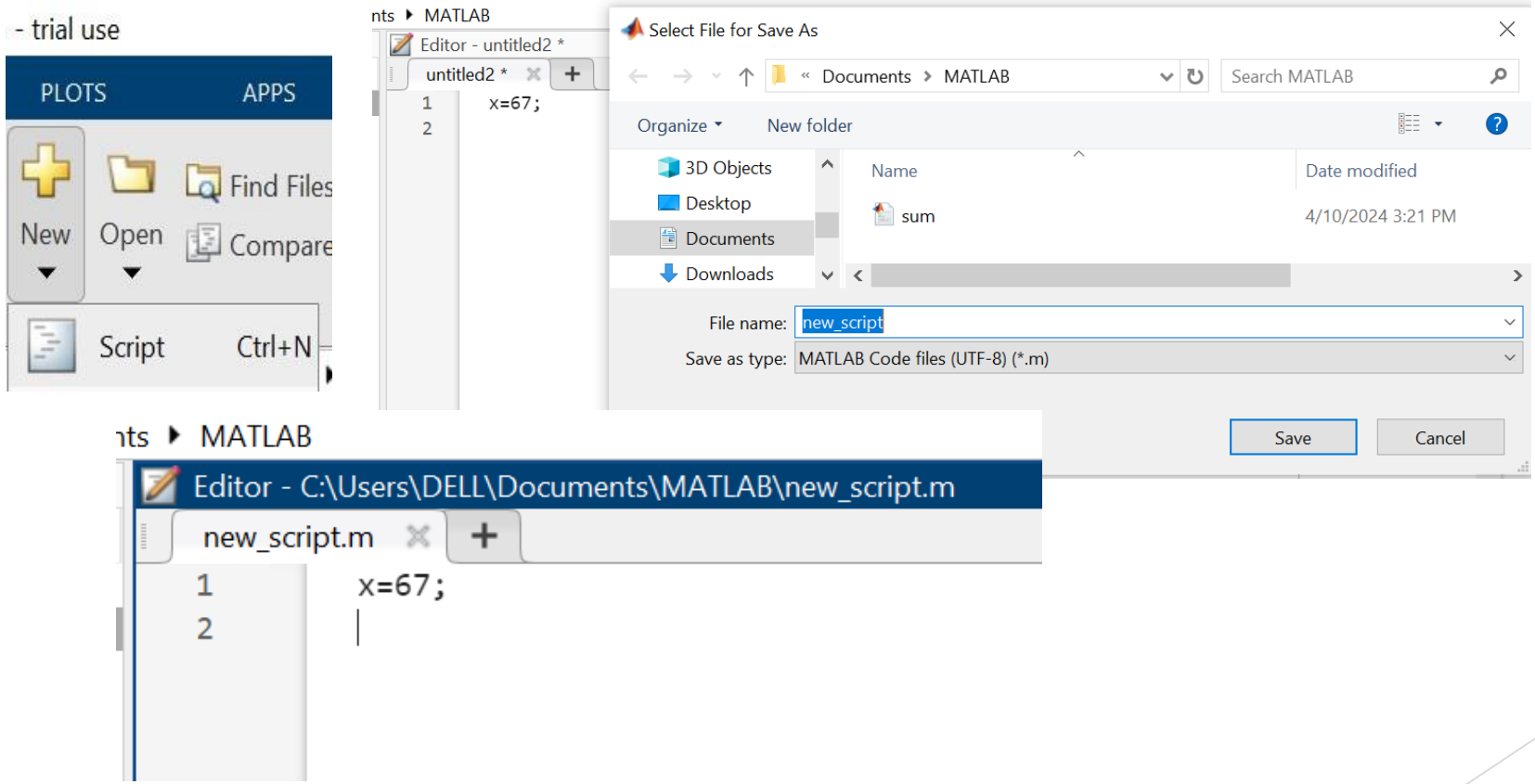
- *4. Command History Window:* This window records all commands previously entered into MATLAB. It functions as a historical record, allowing users to revisit and potentially reuse commands without retyping them. Users can re-execute commands from this window, facilitating efficiency in workflow.

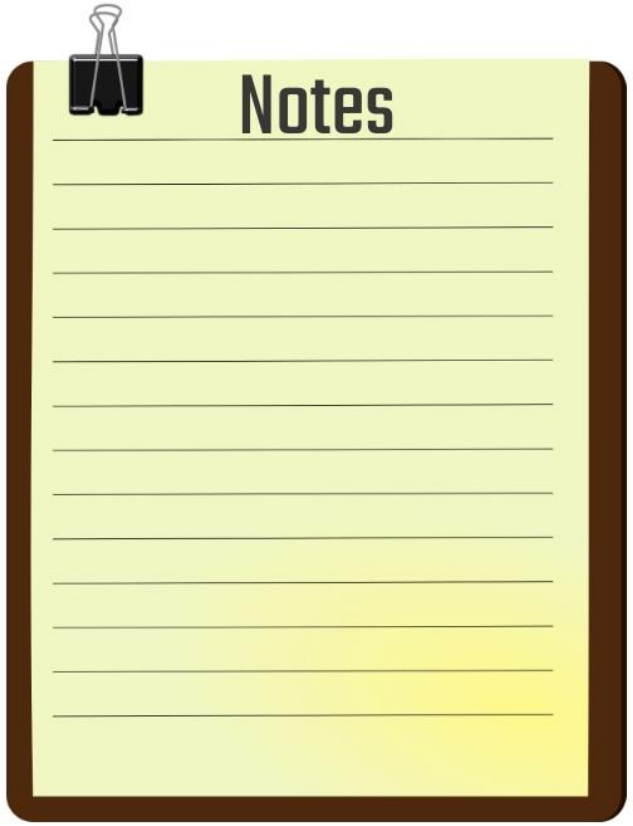




Matlab windows

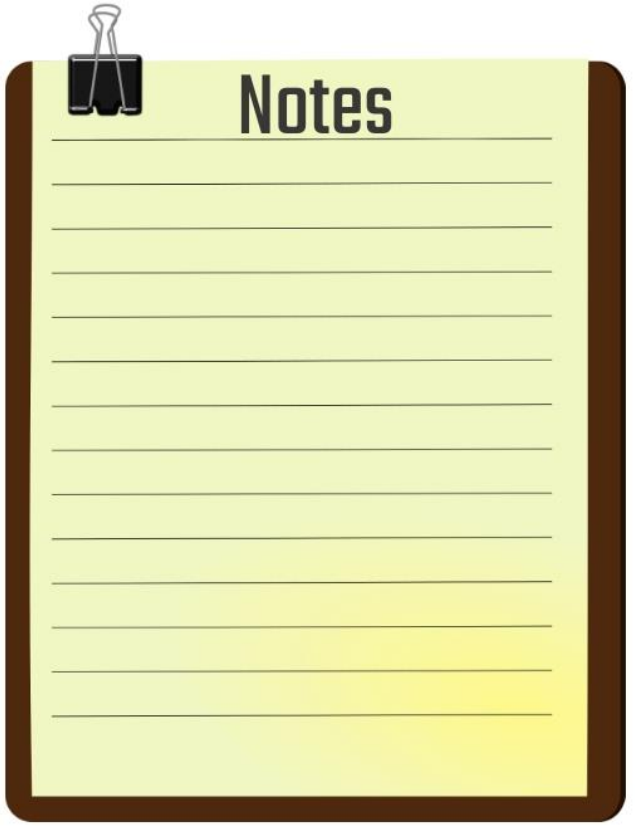
- *5. Editor Window:* The Editor Window offers a platform for writing, editing, and saving complete MATLAB programs. It functions as a text editor tailored for MATLAB code, providing tools for organizing and managing scripts. Additionally, users can run programs directly from this window, streamlining the coding process.



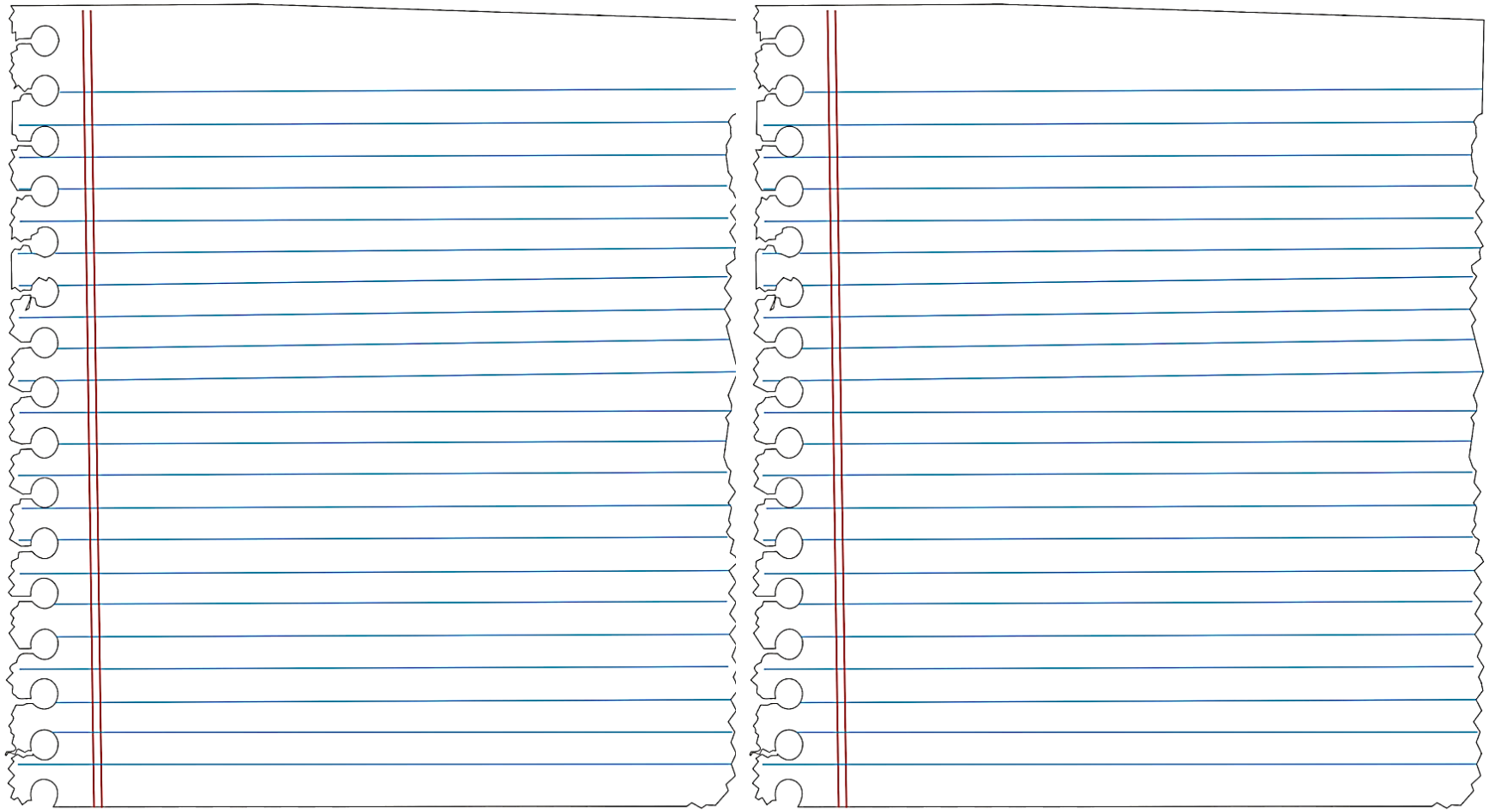


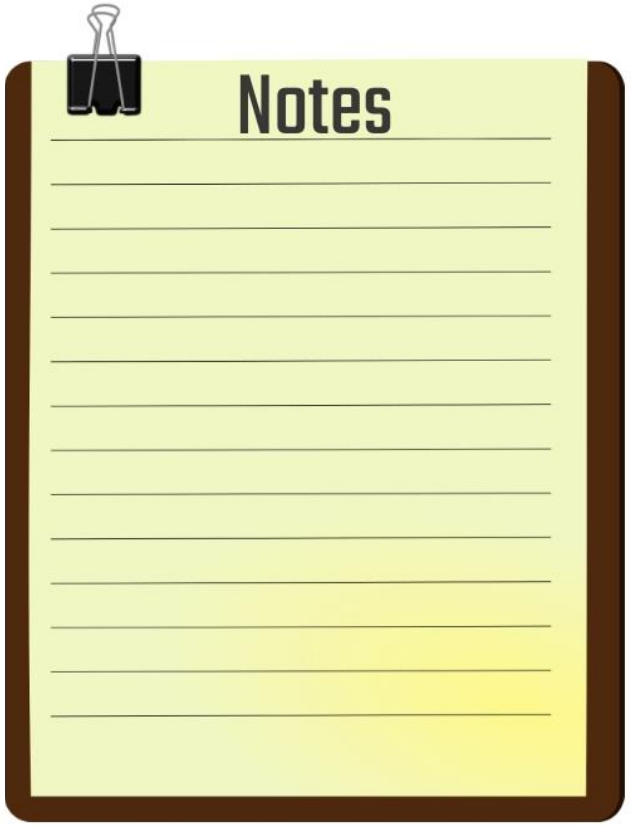
Questions

- 1- Define MATLAB.
- 2- What is the Command Window in MATLAB?
- 3- What is the Workspace Window in MATLAB?
- 4- What is the Help Window in MATLAB?
- 5- What is the Command History Window in MATLAB?
- 6- What is the Editor Window in MATLAB?



Solutions





Computer programming and applications MATLAB-beginner

The first step in Matlab

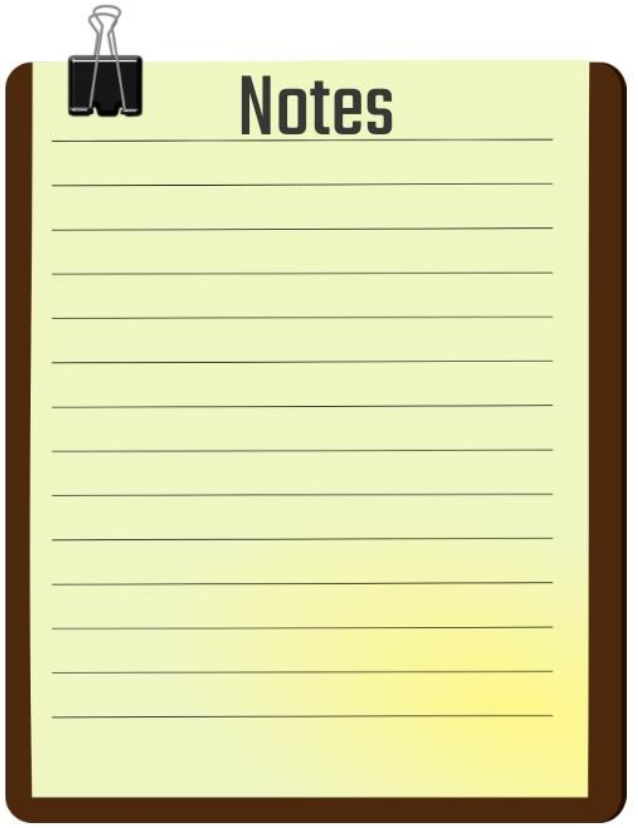
Assist.Lec Aya Abdel Hussein

Constants

- Constants are fixed values that remain unchanged during the execution of a program.
- They are used to represent data that does not change, such as mathematical constants, special values, or fixed parameters.
- Constants are typically directly used in expressions or assigned to variables.

```
Command Window
>> x=4;
>> y=x*6;
>> disp(y);
    24

fx >>
```

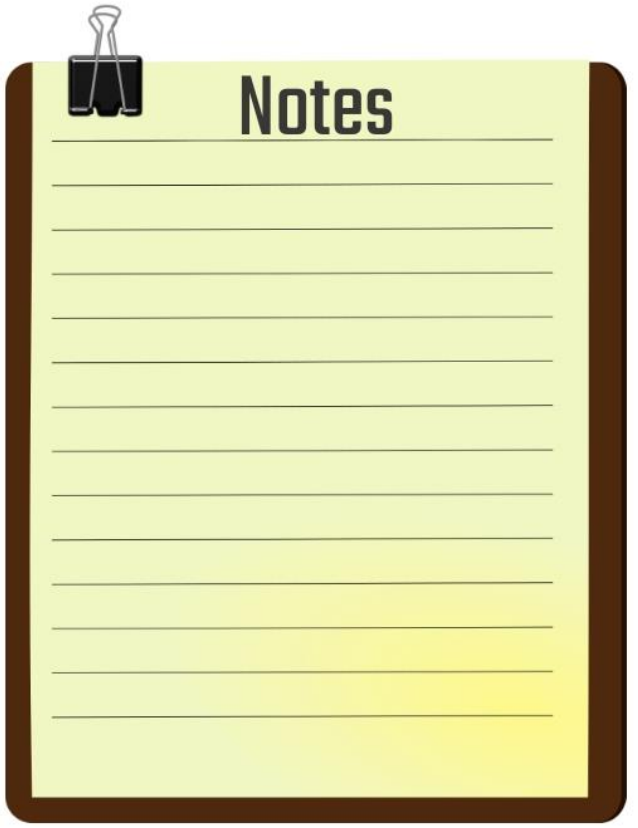
```
Command Window
>> disp(pi);
    3.1416

fx >>
```



Variables

- Variables are symbols that represent values or data.
- They can vary; their values can change during the execution of a program.
- Variables are typically assigned values using the assignment operator '='.
- They can be numeric values, strings, arrays or other data types.

```
Command Window
>> x=5;
>> a='aya';
```



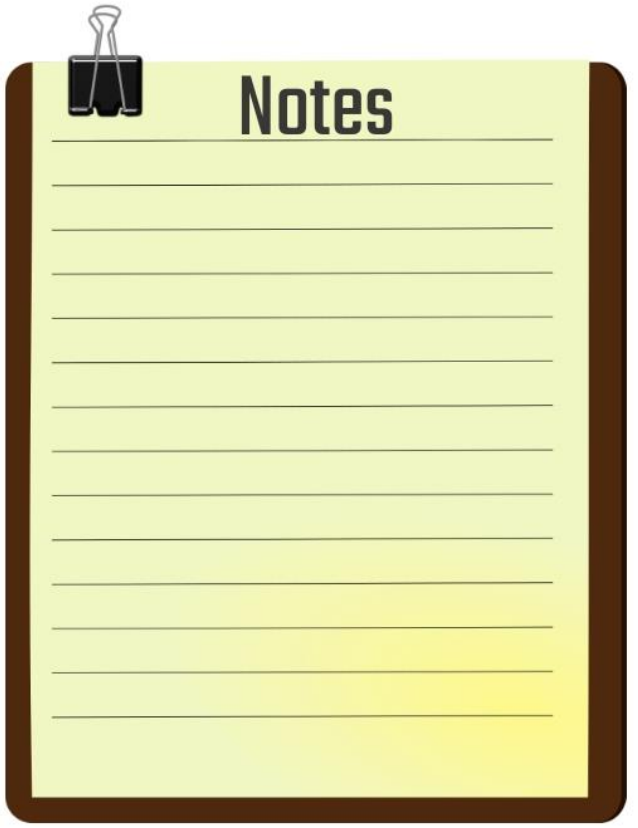
Expressions

- In MATLAB, expressions are combinations of variables, constants, operators, and functions that represent mathematical computations. These expressions are evaluated to produce a result. Here are some examples of expressions in MATLAB:

1-Arithmetic Expressions:

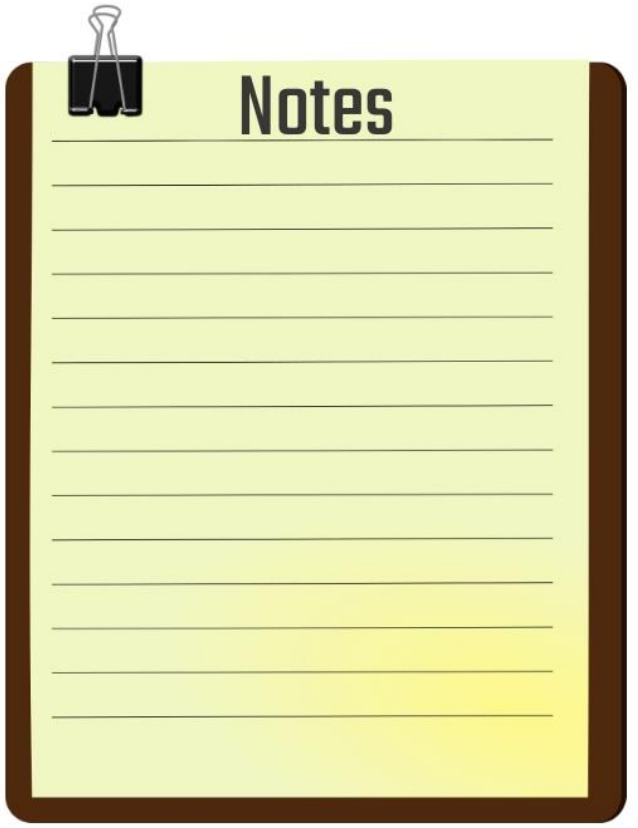
Command Window

```
>> x=8;
>> y=2;
>> z=x*y; disp(z);
    16

>> z=x+y; disp(z);
    10

>> z=x-y; disp(z);
     6

>> z=x/y; disp(z);
     4

>> z=x^y; disp(z);
    64
```



Expressions

write a program in MATLAB that allows the user to input four numbers. The program should then calculate the average of these numbers and display the result.



Expressions

Write a program in Matlab to enter two numbers from the user to perform mathematical operations such as addition, multiplication and subtraction on them.

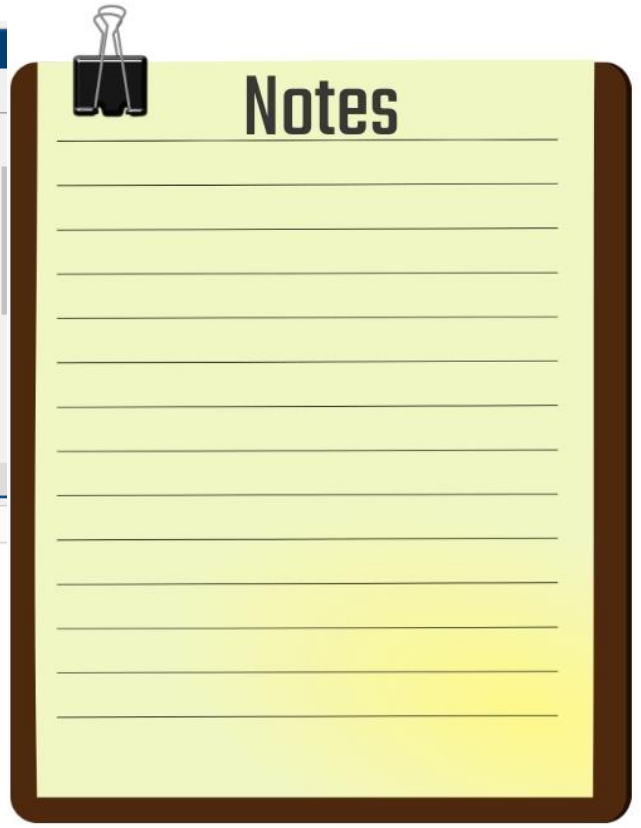
Editor - C:\Users\DELL\Documents\MATLAB\input1_.m

input1_.m

```
num1 = input('Please enter the first number: ');
num2 = input('Please enter the second number: ');
addition = num1 + num2;
subtraction = num1 - num2;
multiplication = num1 * num2;
disp(['Addition of ', num2str(num1), ' and ', num2str(num2), ' results in: ', num2str(addition)]);
disp(['Subtraction of ', num2str(num1), ' and ', num2str(num2), ' results in: ', num2str(subtraction)]);
disp(['Multiplication of ', num2str(num1), ' and ', num2str(num2), ' results in: ', num2str(multiplication)]);
```

Command Window

```
>> input1_
Welcome to the Simple Mathematical Operations Program
Please enter the first number: 2
Please enter the second number: 6
Addition of 2 and 6 results in: 8
Subtraction of 2 and 6 results in: -4
Multiplication of 2 and 6 results in: 12
```

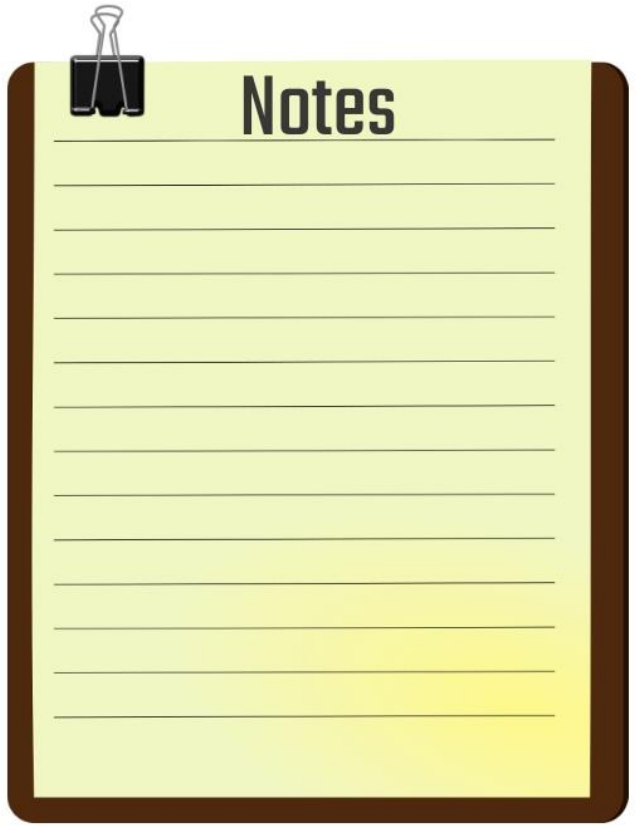


Expressions

2-Relational Expressions:

Command Window

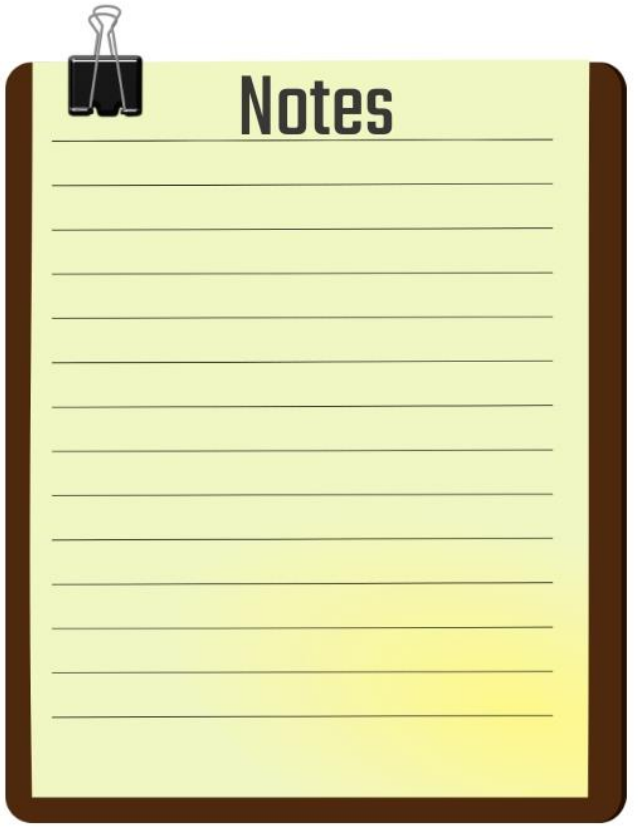
```
>> x=4;
>> y=6;
>> result_greater_than = x > y;
>> disp(result_greater_than);
   0

>> result_greater_than = x < y;
>> disp(result_greater_than);
   1

>> result_greater_than = x ==y;
>> disp(result_greater_than);
   0

>> result_greater_than = x ~=y;
>> disp(result_greater_than);
   1

fx >>
```



Expressions

2-Relational Expressions:

Write a program in MATLAB that prompts the user to enter two numbers and then performs four relational operations on them.

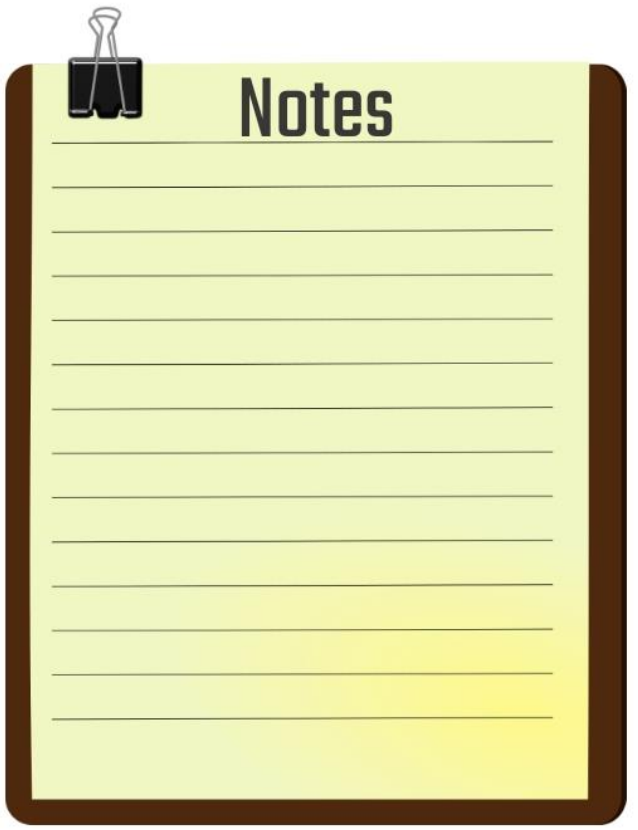


Expressions

3-Logical Expressions:

Command Window

```
>> x = true;
y = false;
result_logical_and = x && y; disp(result_logical_and);
   0

result_logical_or = x || y; disp(result_logical_or);
   1
```



Expressions

3-Logical Expressions:

Write a program in MATLAB that examines two logical values, both set to true, and then performs the logical 'AND' and 'OR' operations on them.

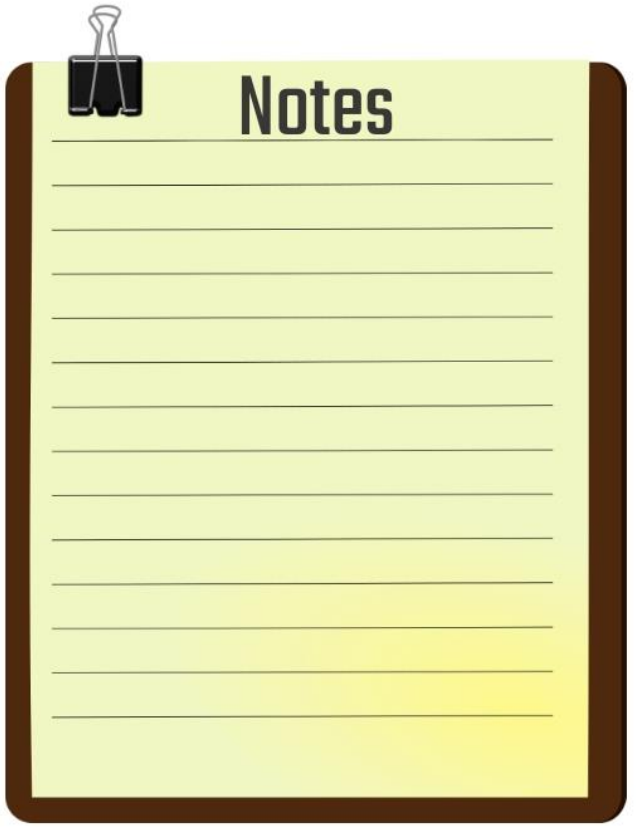
Editor - C:\Users\DELL\Documents\MATLAB\logical.m

logical.m

```
1  logicalValue1 = true;
2  logicalValue2 = true;
3  logicalAndResult = logicalValue1 && logicalValue2;
4  logicalOrResult = logicalValue1 || logicalValue2;
5  disp(['Logical AND of true and true is: ', num2str(logicalAndResult)]);
6  disp(['Logical OR of true and true is: ', num2str(logicalOrResult)]);
7
```

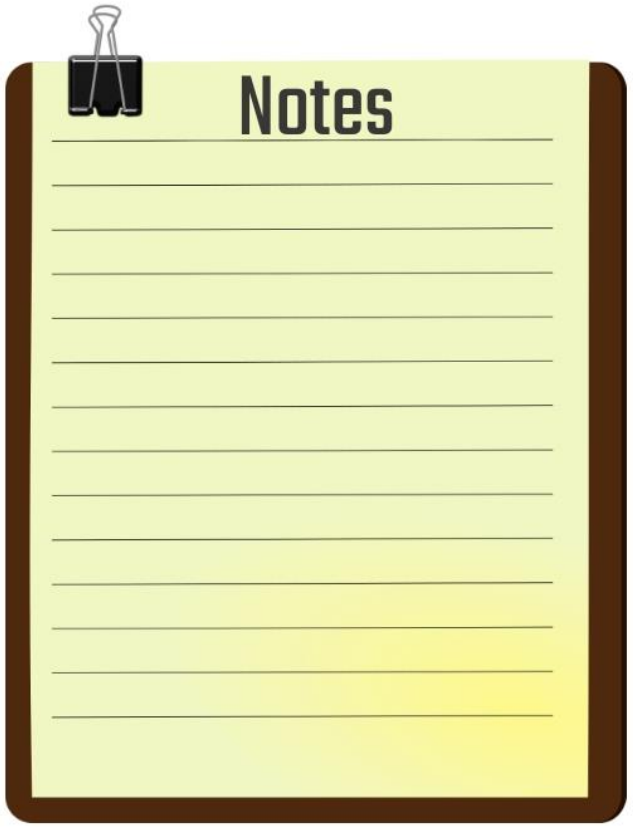
Command Window

```
>> logical
Logical AND of true and true is: 1
Logical OR of true and true is: 1
fx >>
```

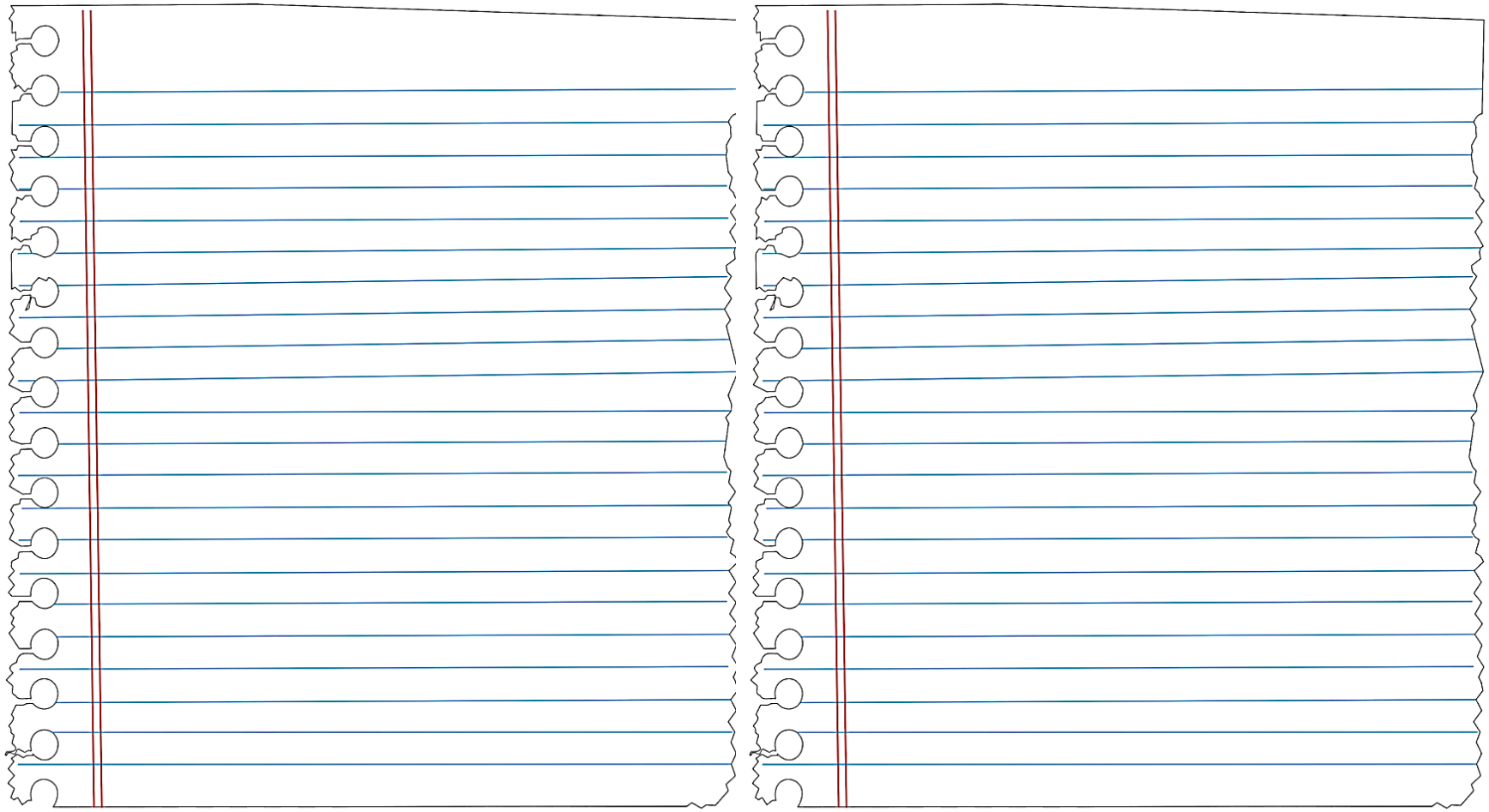


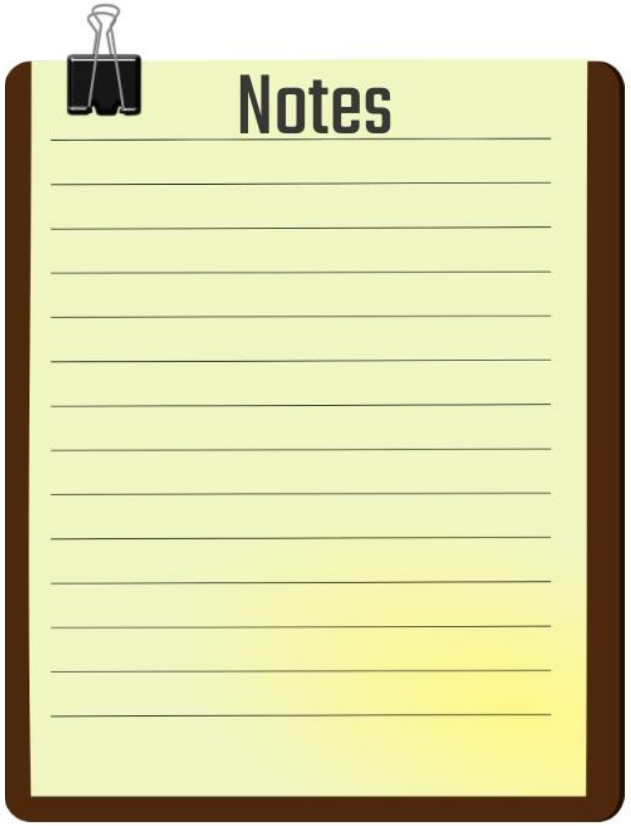
Questions:

1. Define the variables.
2. Define the constants.
3. What are expressions? Provide an example of each type.
4. Write a program in MATLAB to solve the following equations:
   - $R = A \times B - \frac{C}{D}$
   - $S = A - B \times C + D$
5. Given $x = 25$ and $y = 33$, calculate the value of z in MATLAB using the following equation:
   - $Z = x - 3 + 88 \times y$
6. Let $v(t)$ represent the speed of a body moving in a straight line, where $v(t) = 6 - 3t + 2$. Calculate the speed in MATLAB when $t = 2$.



Solutions





Computer programming and applications MATLAB-beginner

Arrays

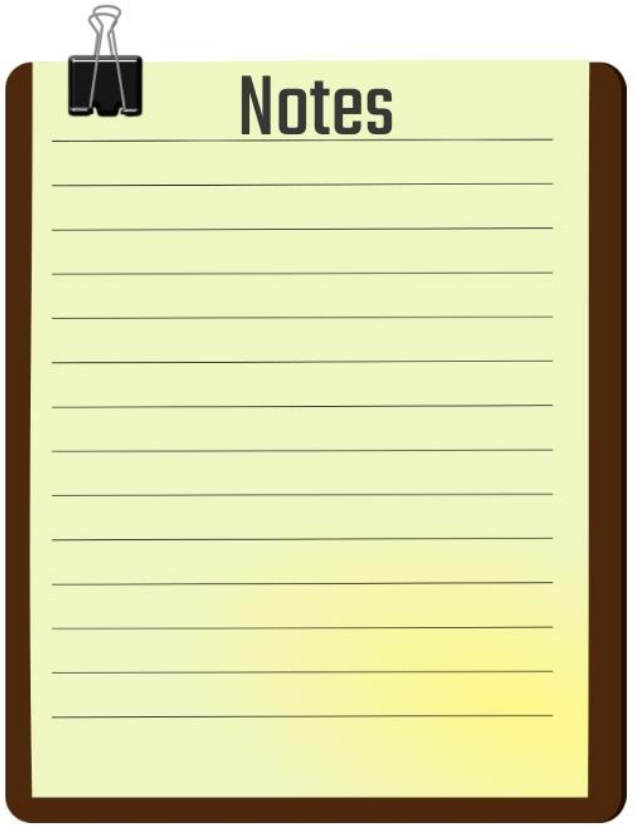
Assist.Lec Aya Abdel Hussein

matrix

- A matrix is a mathematical concept that represents a rectangular array of numbers, symbols, or expressions arranged in rows and columns. Each element in a matrix is identified by its row and column indices. Matrices are widely used in linear algebra and various fields such as physics, engineering, computer graphics, and economics.

Properties of Matrices:

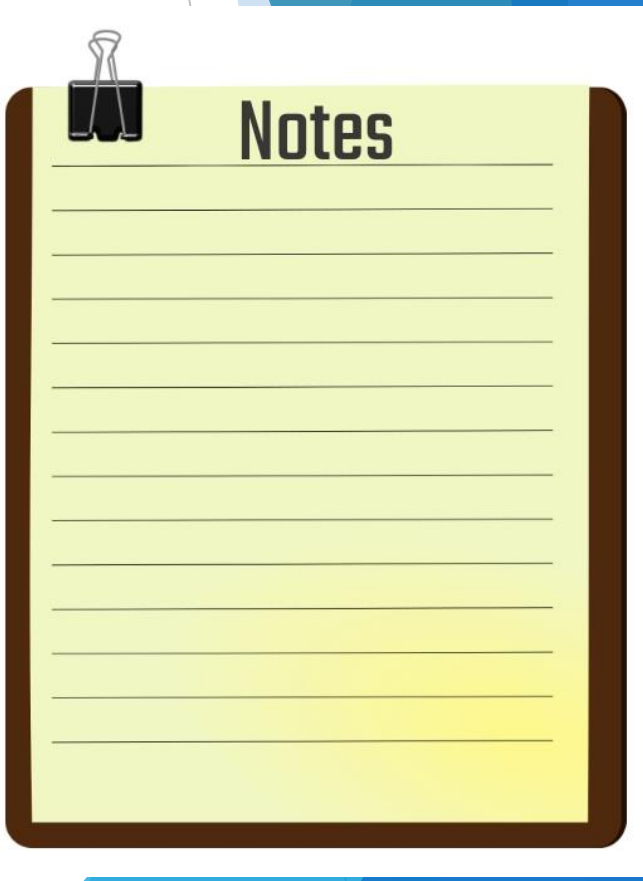
- 1. Size and Dimensions: A matrix has a defined number of rows and columns, which determine its dimensions or size.
- 2. Elements: The individual entries in a matrix are called elements. Each element is identified by its row and column indices.
- 3. Types of Matrices: Matrices can be square (same number of rows and columns) or rectangular. They can also be classified based on other properties like symmetry, diagonal dominance, etc.
- 4. Operations:Matrices support operations such as addition, subtraction, scalar multiplication, matrix multiplication,and transpose.



Entering Matrices

Manual Entry

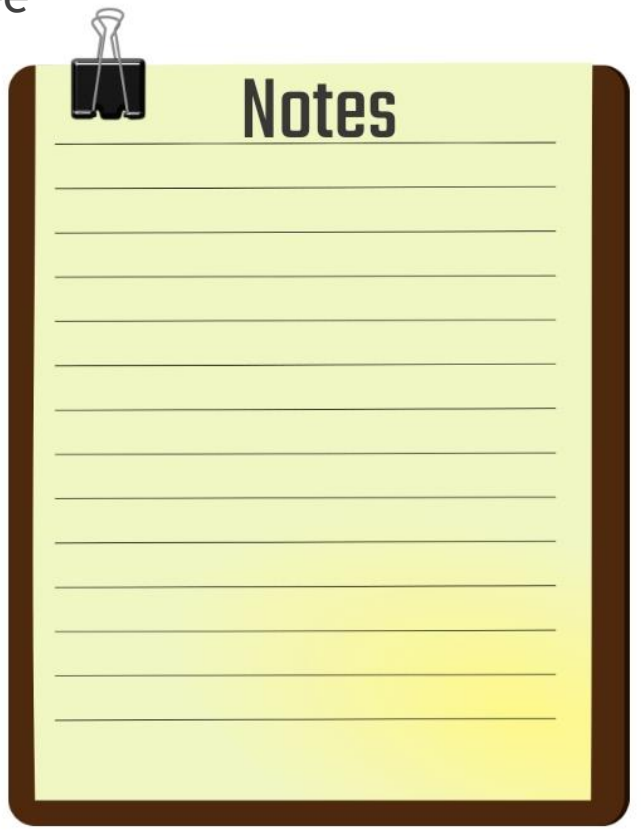
```
Command Window
>> A=[1 2 3 4 6 8]; disp(A);
     1     2     3     4     6     8

>> A=[3, 9, 0, 7]; disp(A);
     3     9     0     7

fx >>

>> A = [1, 8;8, 9];disp(A);
     1     8
     8     9
```



Entering Matrices

Manual Entry

Write MATLAB code to manually enter a matrix consisting of four rows and three columns.



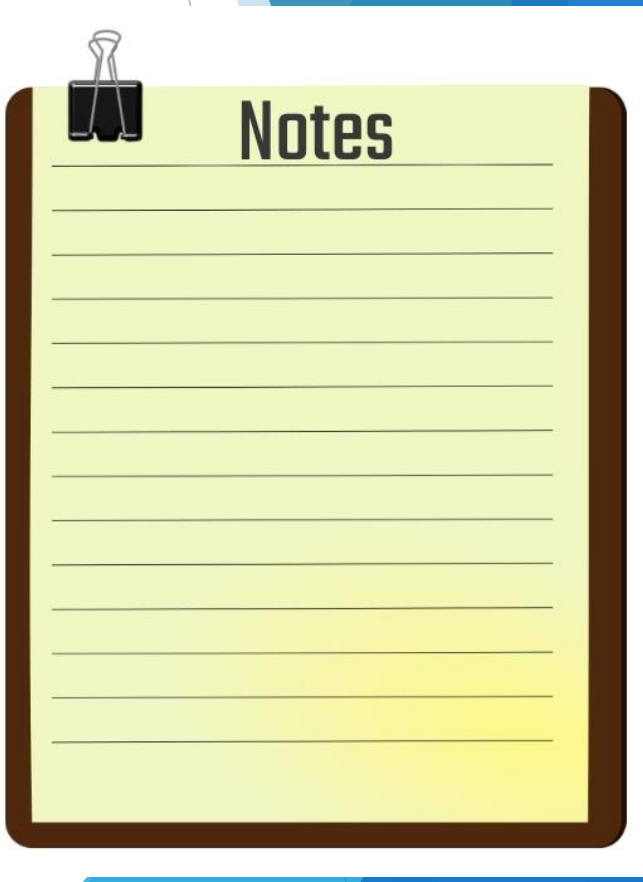
Entering Matrices

Using the zeros, ones, or rand Functions:

```
>> B = zeros(3, 4);disp(B);
     0     0     0     0
     0     0     0     0
     0     0     0     0
```

```
>> C = ones(2, 2); disp(C);
     1     1
     1     1
```

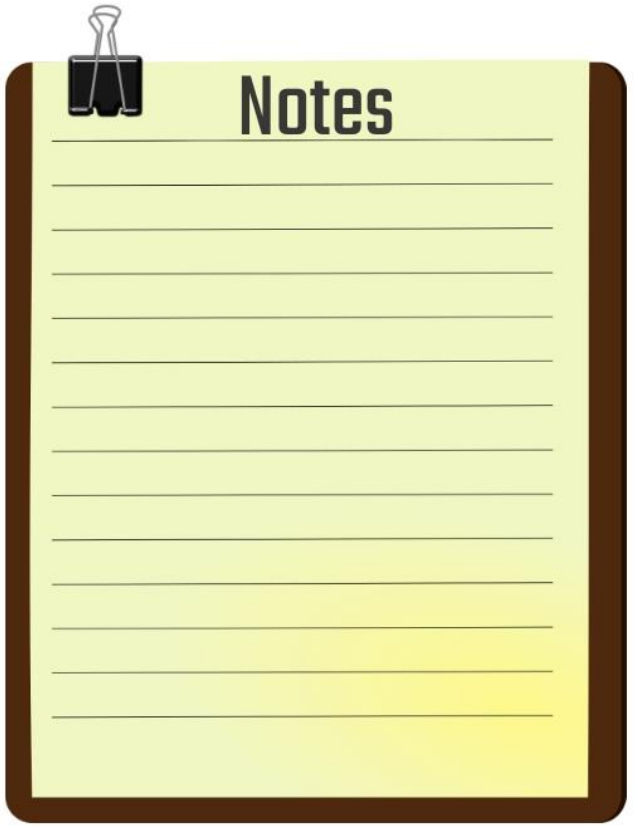
```
>> D = rand(3, 2); disp(D);
    0.8147    0.9134
    0.9058    0.6324
    0.1270    0.0975
```



Entering Matrices

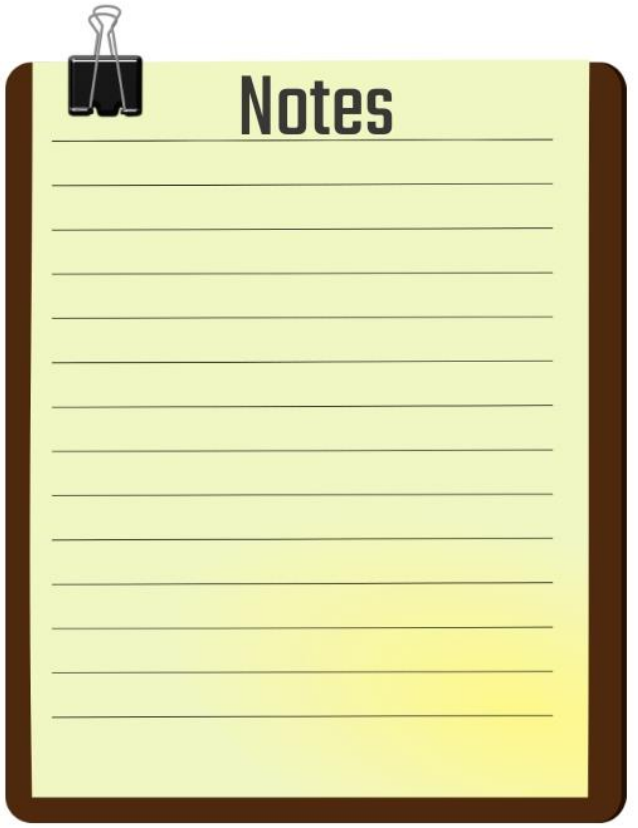
Using the zeros, ones, or rand Functions:

Write a program in MATLAB that generates a 4x4 matrix in three ways: the first matrix should be composed of zeros, the second of ones, and the third of random decimal values.



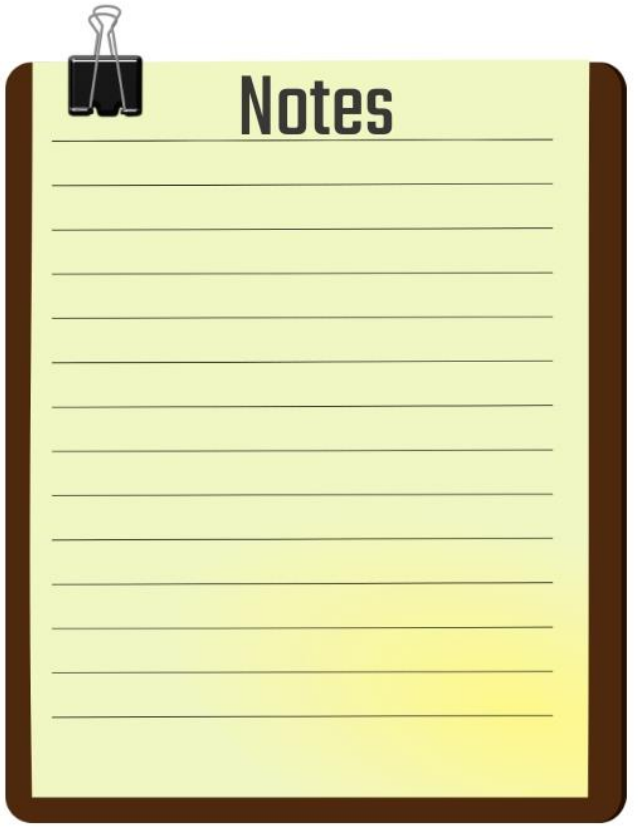
Deleting Rows or Columns.

```
Command Window
>> A = [1, 2, 3; 4, 5, 6; 7, 8, 9;0 5 2 ];
>> disp(A);
     1     2     3
     4     5     6
     7     8     9
     0     5     2

>> A(:,2)=[];
>> disp(A);
     1     3
     4     6
     7     9
     0     2

>> A(3,:)=[]; disp(A);
     1     3
     4     6
     0     2

fx >>
```



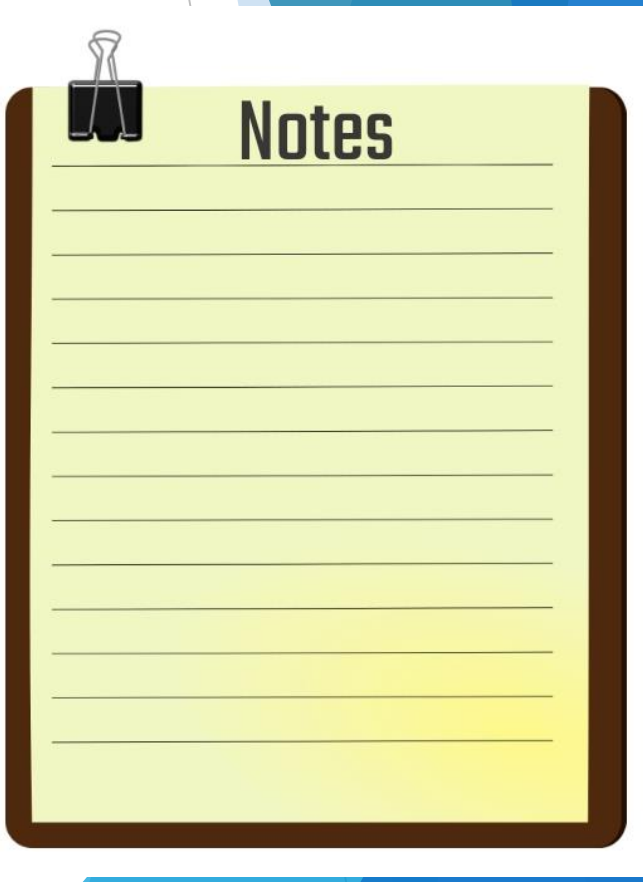
Transpose

```
Command Window
>> A = [1, 2, 3; 4, 5, 6; 7, 8, 9]; disp(A);
     1     2     3
     4     5     6
     7     8     9

>> A_transposed = transpose(A); disp(A_transposed);
     1     4     7
     2     5     8
     3     6     9

fx >>
```



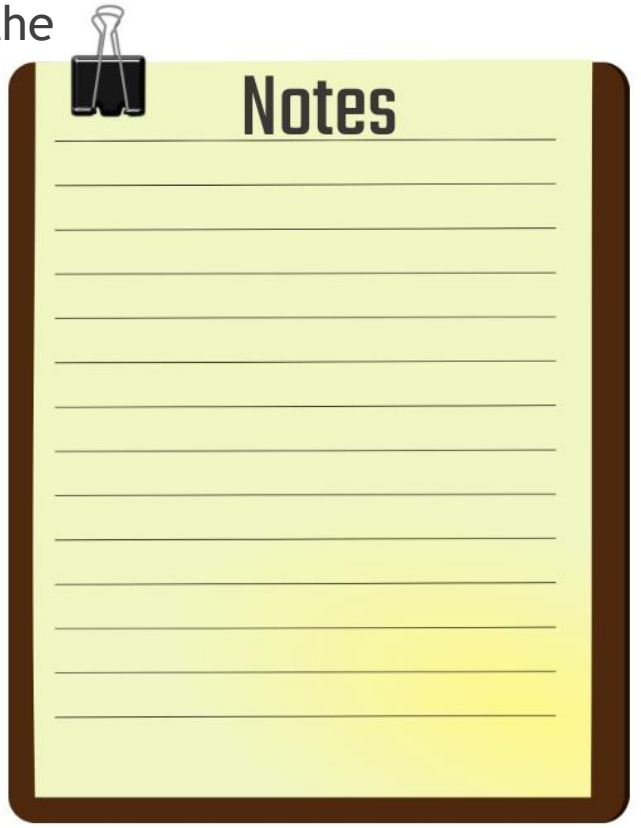
Change the value of an element in an array

```
Command Window
>>  A = [1, 0, 3; 4, 5, 6; 7, 3, 9]; disp(A);
     1     0     3
     4     5     6
     7     3     9

>> A(2,3)=10; disp(A);
     1     0     3
     4     5    10
     7     3     9

fx >>
```



Questions

- 1- Define the matrix and explain its most important properties.
- 2- Write MATLAB code to manually enter a matrix consisting of four rows and three columns. Then perform a transpose between the rows and columns, delete the second row, and replace the first element in the array with the number 100.

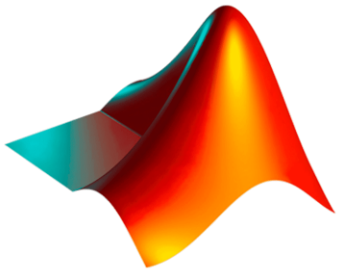


Solutions

Notes

Computer programming and applications MATLAB-beginner

Built in functions



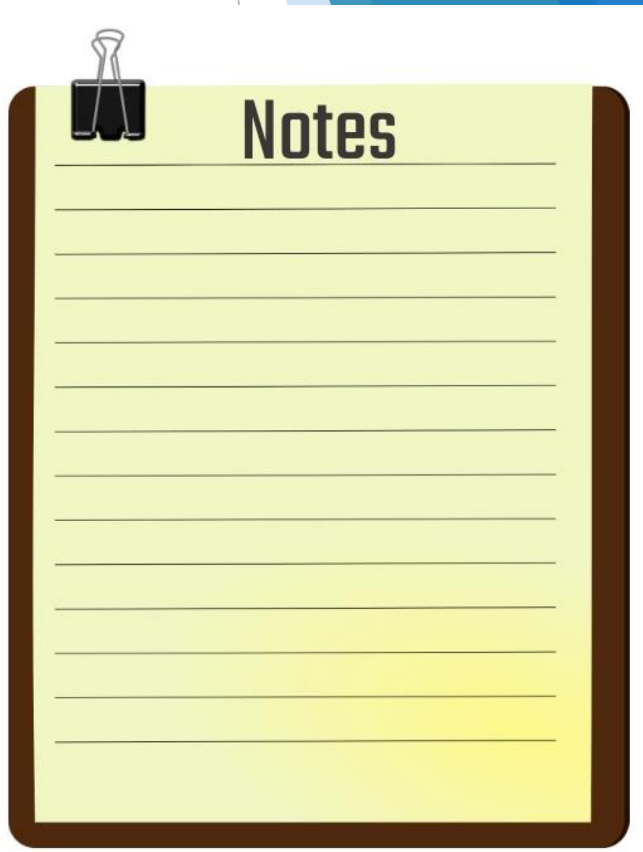
MATLAB®

Assist.Lec Aya Abdel Hussein

Built-in functions

- Built-in functions: are predefined functions that come with the MATLAB environment, designed to perform a variety of specific tasks in areas including mathematics, statistics, engineering, and graphics. These functions are ready to use and help to efficiently perform complex operations without the need for writing extensive code from scratch.

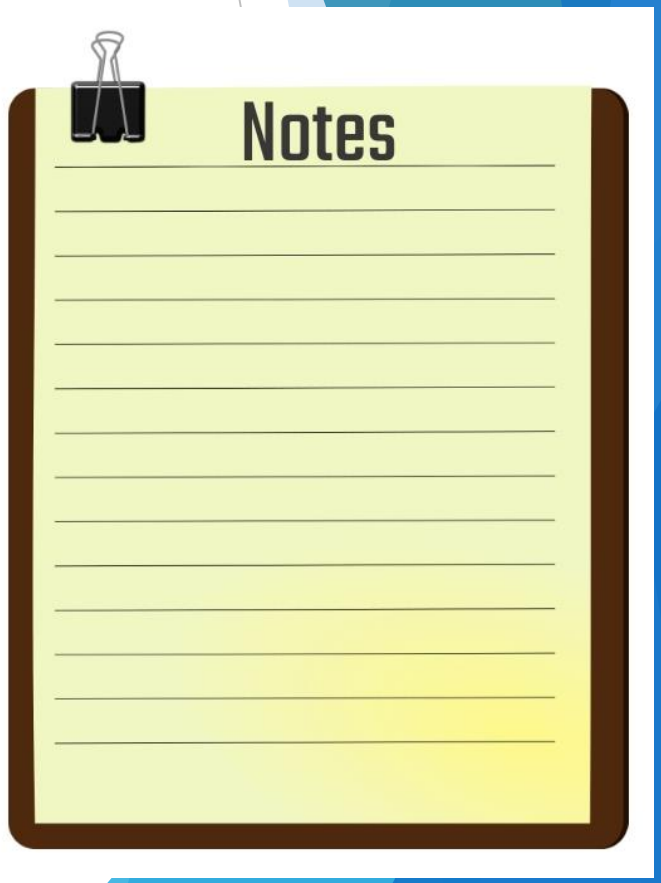
Type	Description	Example
Mathematical Functions	Functions for mathematical operations	`result = sin(pi/3);`
		`result = log10(100);`
Statistical Functions	Functions for statistical analysis	`average = mean([1, 2, 3]);`
		`med = median([1, 3, 3, 6, 7]);`
Plotting Functions	Functions for creating plots and visualizations	`plot(0:pi/100:2*pi, sin(0:pi/100:2*pi));`
		`hist(randn(1000, 1), 25);`
File I/O Functions	Functions for input/output operations	`load('data.mat');`
		`save('data.mat', 'data');`



Built-in functions

Built in functions Always shorten and simplify programming processes

```
Command Window
>> x=1; y=2; z=3;
 sum=(x+y+z);
 average=sum/3;
 disp(average);
     2

>> average=mean([1, 2, 3]); disp(average);
     2

fx >>
```

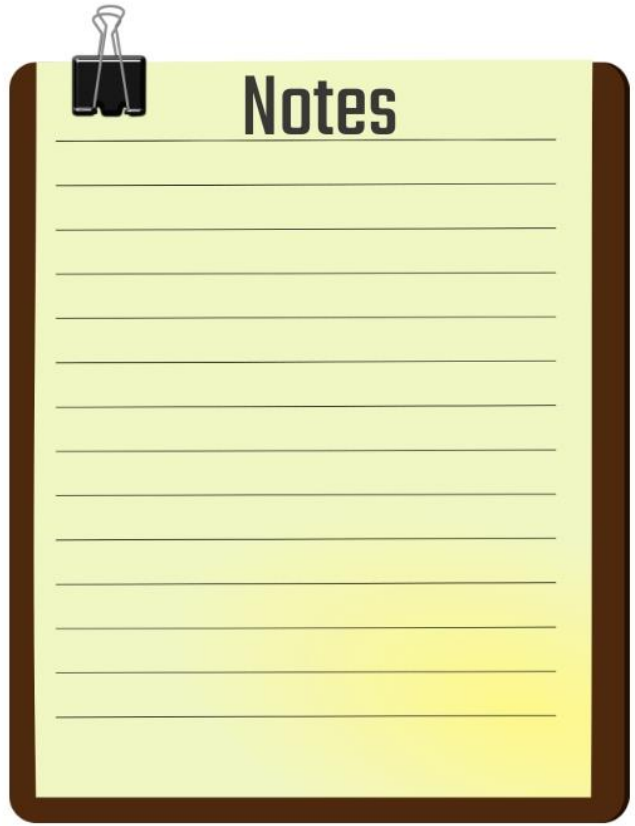


Basic Matrix Functions

- Basic matrix functions in MATLAB are fundamental operations used for manipulating matrices and performing common tasks.
- *1- sum:* Computes the sum of all elements in a matrix.

Command Window

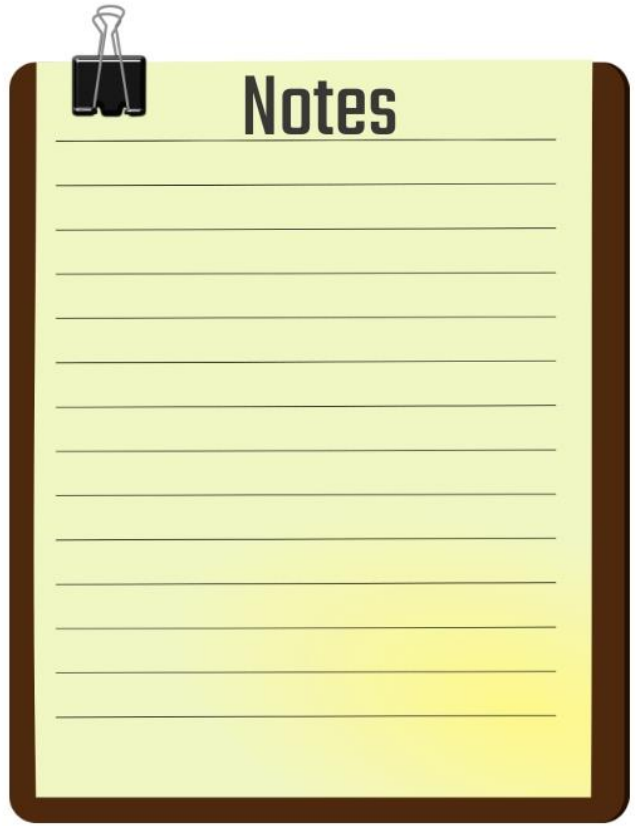
```
>> A = [1, 2, 3; 4, 5, 6; 7, 8, 9];
total = sum(A(:)); disp(total);
    45
```



Basic Matrix Functions

- *2-max:* Finds the maximum value in a matrix.
- *3-min*: Finds the minimum value in a matrix.

```
Command Window
>> A = [3, 7, 2; 9, 4, 6];
max_val = max(A(:));disp(max_val);
     9

>> min_val = min(A(:));disp(min_val);
     2

fx >>
```



Basic Matrix Functions

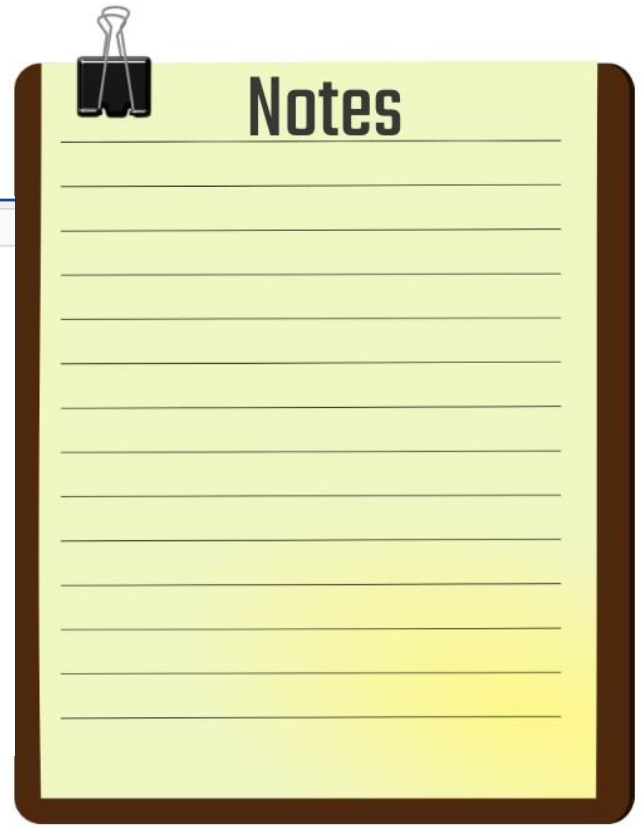
- Write a program in MATLAB that finds the largest and smallest elements of a 3x3 matrix and multiplies them together.

max_x_min.m

```
matrix =[2 3 6; 5 8 7; 2 9 10];
disp('Original Matrix:');
disp(matrix);
max_element = max(matrix(:));
min_element = min(matrix(:));
result = max_element * min_element;
disp('Largest Element:');
disp(max_element);
disp('Smallest Element:');
disp(min_element);
disp('Product of Largest and Smallest Elements:');
disp(result);
```

Command Window

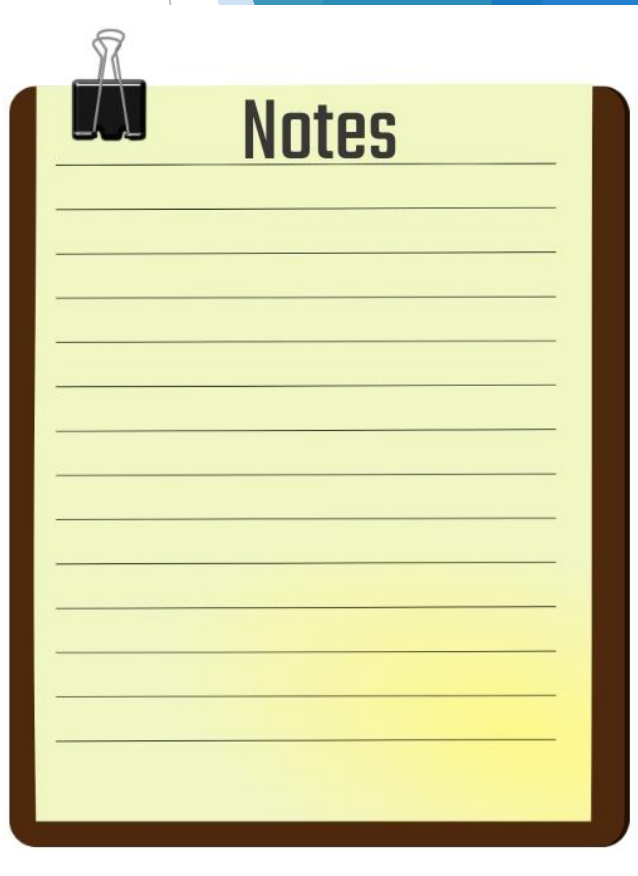
```
Original Matrix:
     2     3     6
     5     8     7
     2     9    10

Largest Element:
    10

Smallest Element:
     2

Product of Largest and Smallest Elements:
    20
```



Basic Matrix Functions

- *4- mean:* Computes the mean of all elements in a matrix.

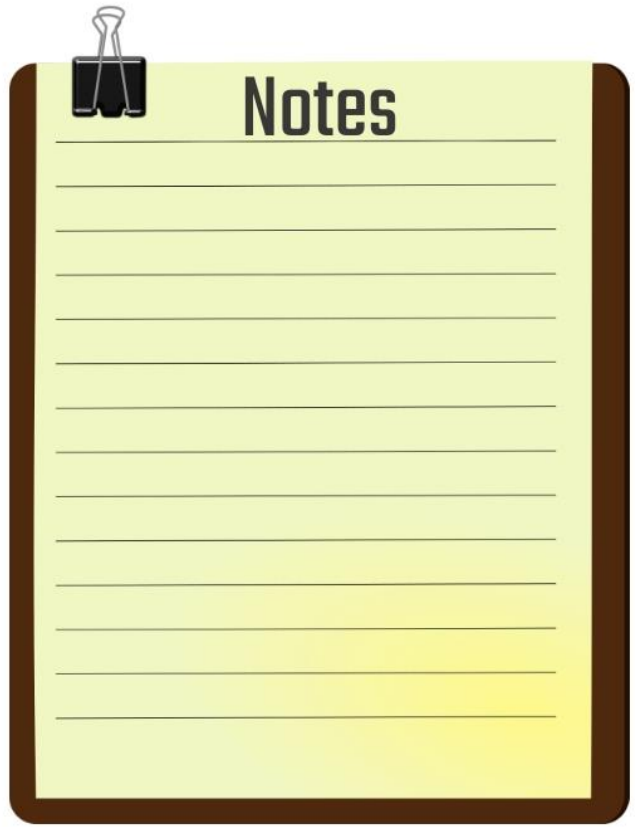
```
Command Window
>> A = [1, 2, 3; 4, 5, 6; 7, 8, 9];
avg = mean(A(:)); disp(avg);
     5

fx >>
```



Basic Matrix Functions

- *5- magic:* Generates a magic square matrix of a given size.
- A magic square is a square matrix in which the sums of the elements in each row, each column, and both main diagonals are the same.

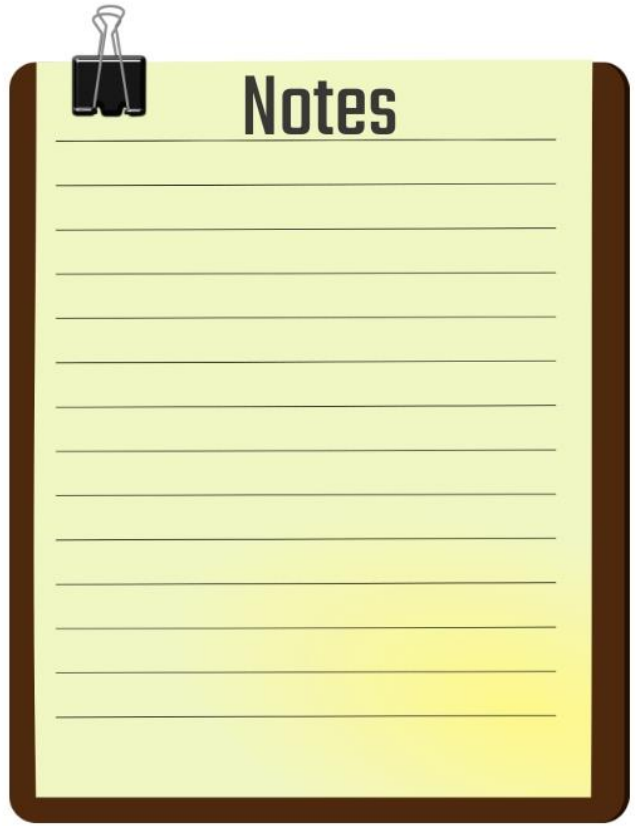
```
>> magic_square = magic(3);disp(magic_square);
     8     1     6
     3     5     7
     4     9     2

>> magic_square = magic(4);disp(magic_square);
    16     2     3    13
     5    11    10     8
     9     7     6    12
     4    14    15     1

>> magic_square = magic(5);disp(magic_square);
    17    24     1     8    15
    23     5     7    14    16
     4     6    13    20    22
    10    12    19    21     3
    11    18    25     2     9
```



Basic Matrix Functions

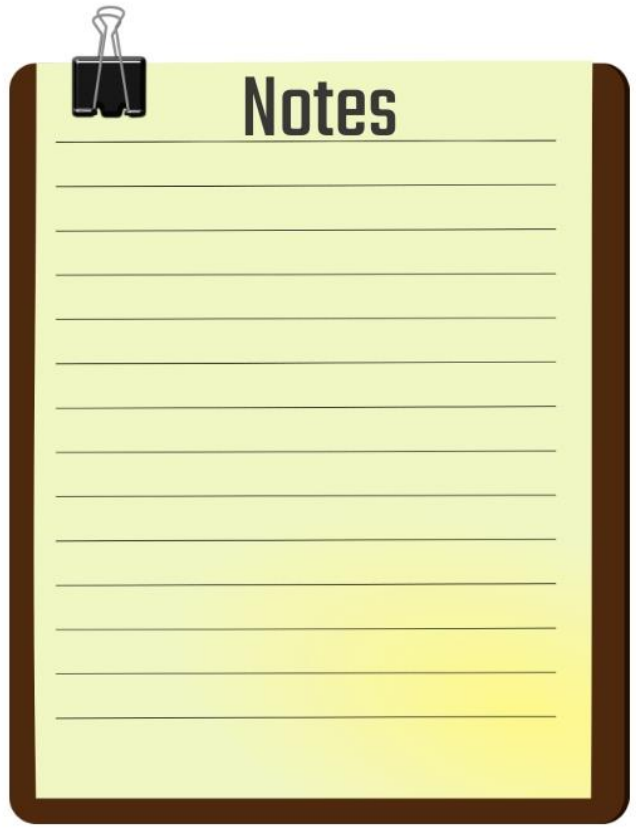
- *6- diag:* Extracts the diagonal elements of a matrix or constructs a diagonal matrix from a vector.

```
Command Window
>> A = [1, 2, 3; 4, 5, 6; 7, 8, 9];
diagonal = diag(A); disp(A);
     1     2     3
     4     5     6
     7     8     9

>> disp(diagonal);
     1
     5
     9
```



Basic Matrix Functions

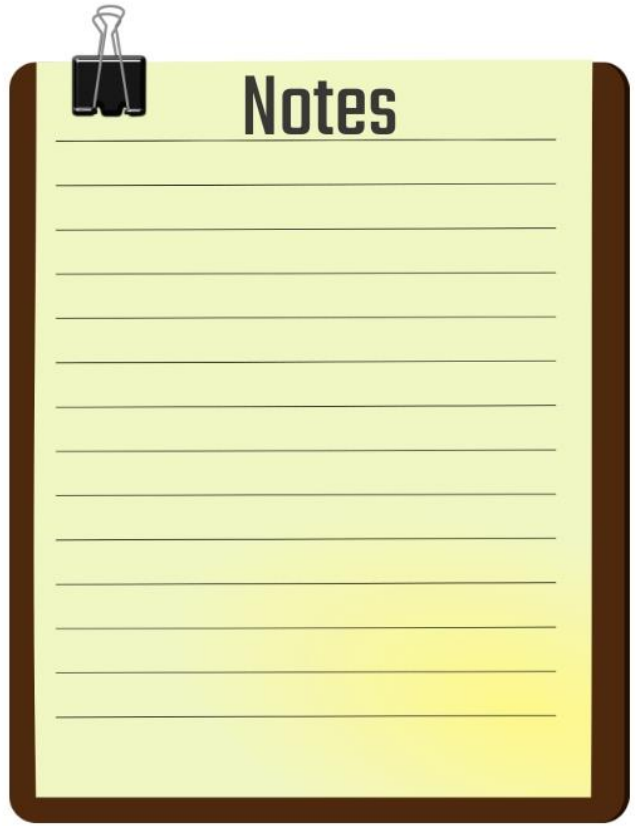
Create a magic matrix in MATLAB, then find its average and major diagonal.



Basic Matrix Functions

- *7- length:* Returns the length of the largest array dimension.
- *8- size:* Returns the size of a matrix.

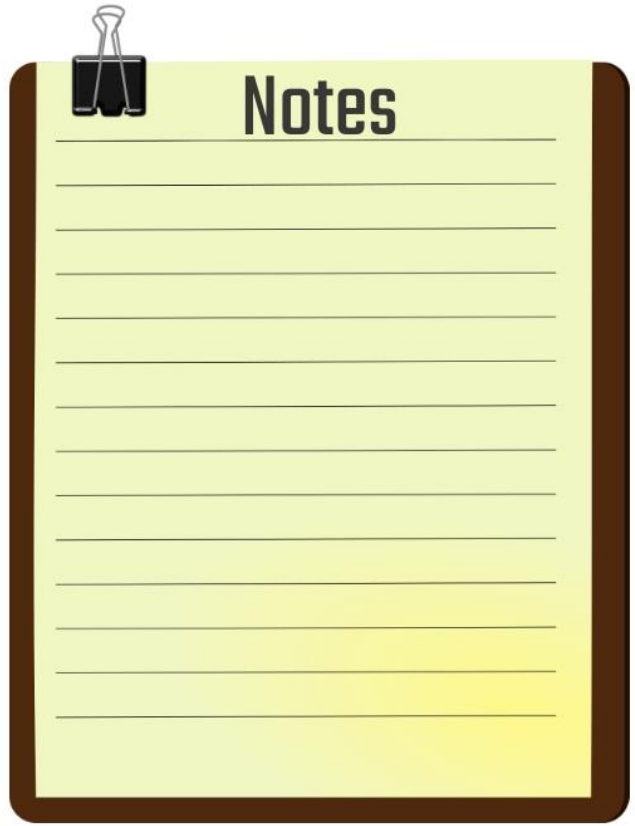
```
Command Window
>> A = [1, 2, 3; 4, 5, 6];
len = length(A); disp(len);
     3

>> sz = size(A); disp(sz);
     2     3

fx >>
```



Basic Matrix Functions

- *9- median:* Computes the median value of all elements in a matrix.
- *10- prod:* Computes the product of all elements in a matrix.

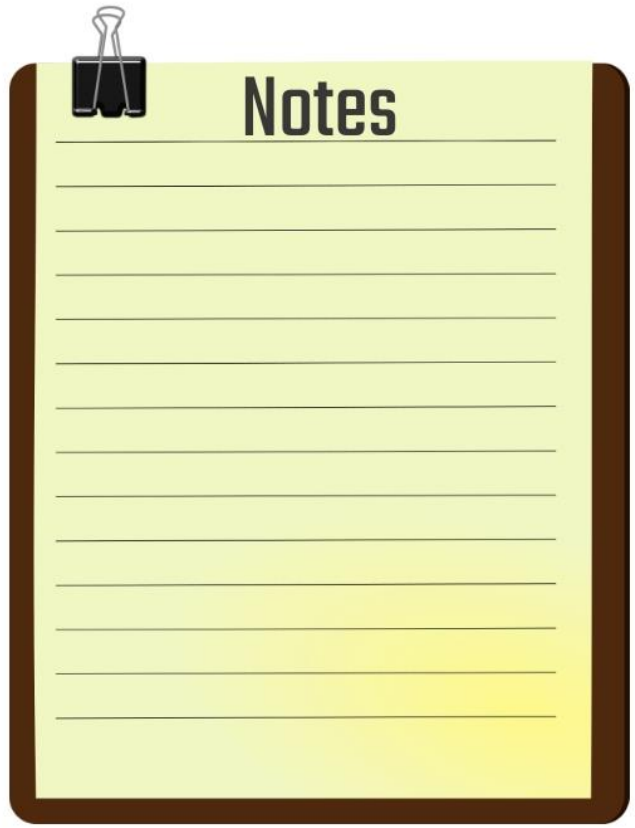
```
Command Window
>> A = [2, 3, 7; 8, 5, 4; 1, 6, 9];
med = median(A(:)); disp(med);
     5

>> A = [1, 2, 3; 4, 5, 6; 7, 8, 9];
product = prod(A(:)); disp(product);
      362880

fx >>
```



Basic Matrix Functions

- *11- sort:* Sorts the elements of a matrix in ascending order.

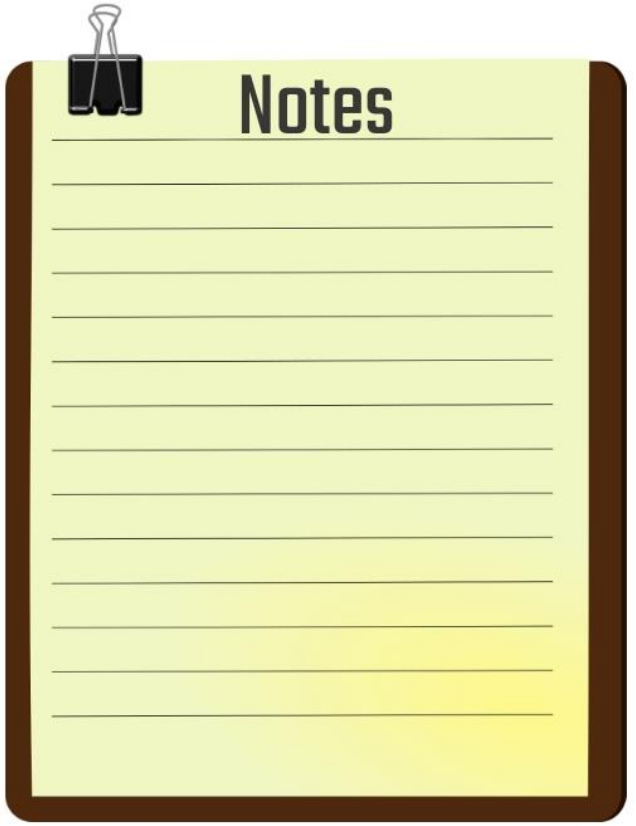
```
Command Window
>> A = [3, 1, 5; 4, 2, 6];
sorted_A = sort(A(:)); disp(sorted_A);
     1
     2
     3
     4
     5
     6
```



Questions

- Define built-in functions. Then, in MATLAB, create a 4x3 matrix and perform the following operations:

- 1. Find the sum of all matrix values.
- 2. Determine the largest and smallest element in the matrix.
- 3. Calculate the average of all array elements.
- 4. Compute the main diagonal of the matrix.
- 5. Find the length of the array (total number of elements).
- 6. Determine the size of the matrix (rows x columns).
- 7. Find the median of all matrix elements.
- 8. Calculate the product of all matrix elements.



Solutions

Notes

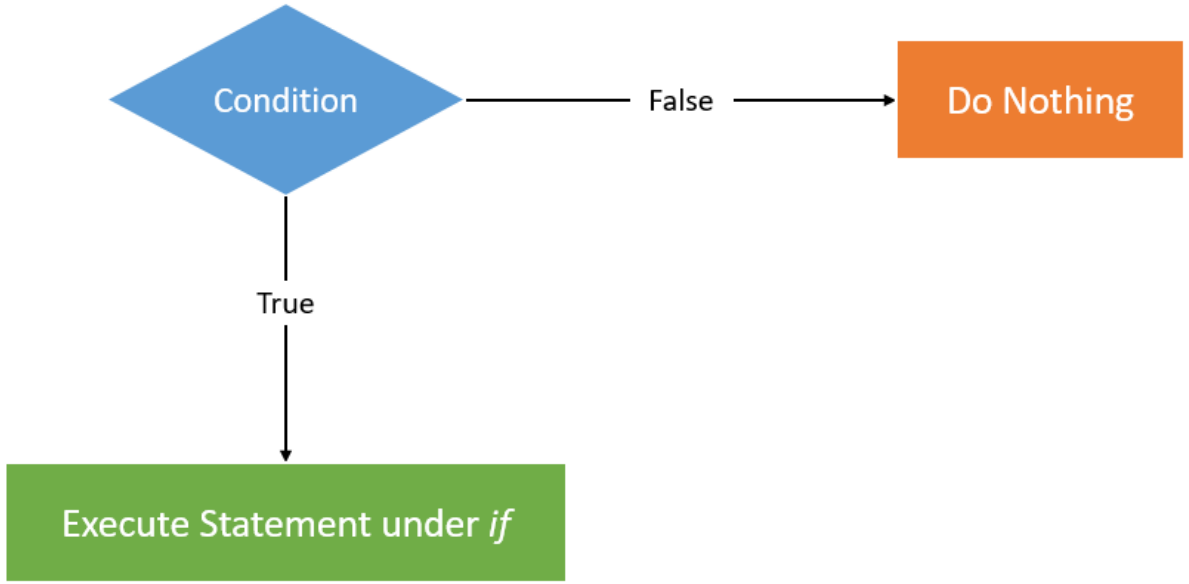
Computer programming and applications MATLAB-beginner

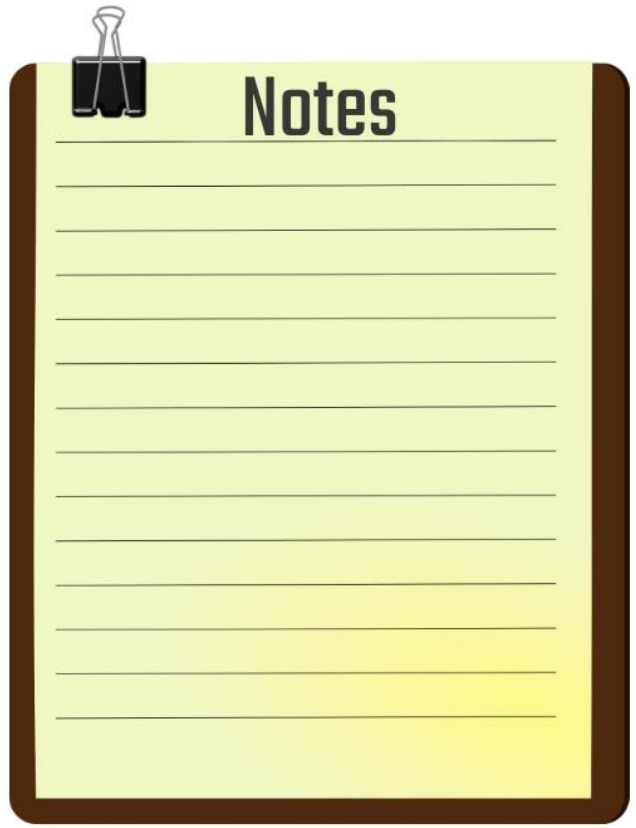
Control Statements

Assist.Lec Aya Abdel Hussein

Conditional statements

Conditional statements, also known as control statements, are programming constructs that allow you to execute different blocks of code based on certain conditions or criteria. They are fundamental to controlling the flow of execution in a program.





Conditional statements: If

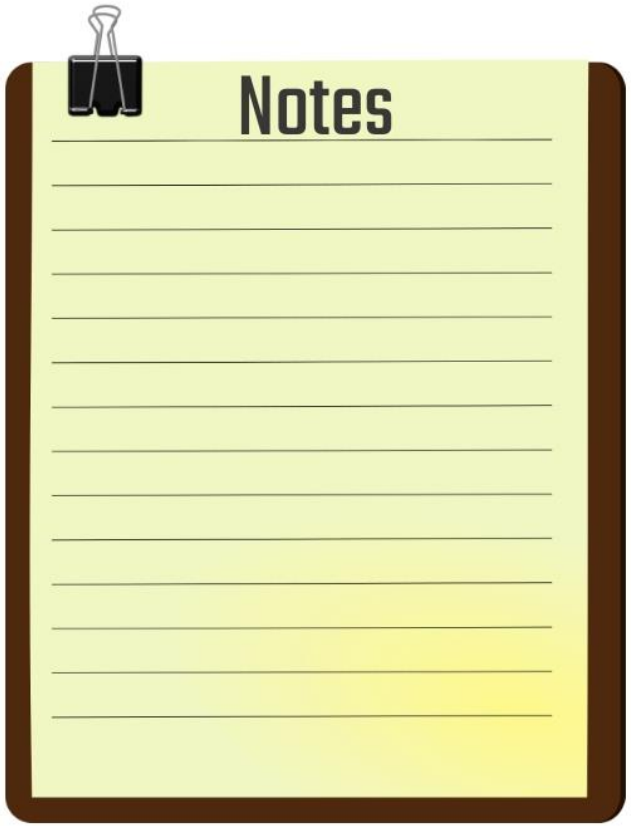
An if-end statement is the simplest decision-making statement. It decides whether a particular block of code has to be executed or not, based on the given boolean condition. Only when the given condition is true, it executes the statements inside the block otherwise not.

Editor - C:\Users\DELL\Documents\MATLAB\if6.m

if6.m

```
1   x = 10;
2   if x > 5
3       disp('x is greater than 5');
4   end
```

```
x is greater than 5
>>
```



Conditional statements: If

Write a MATLAB program that classifies a number as positive, negative, or zero using only if statements.

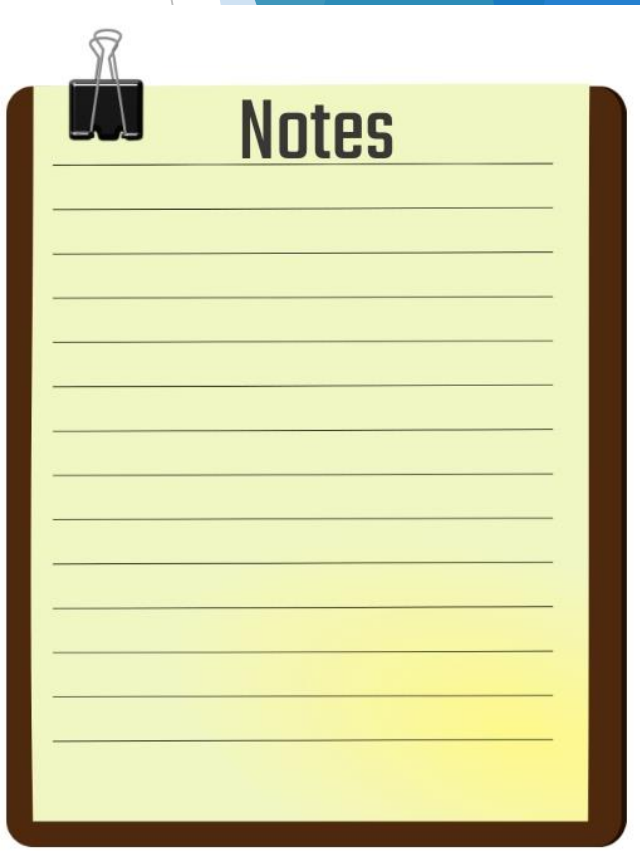
Editor - C:\Users\DELL\Documents\MATLAB\if26.m

if26.m

```
1   num = input('Enter a number: ');
2
3   if num > 0
4       disp([num2str(num) ' is positive.']);
5   end
6
7   if num < 0
8       disp([num2str(num) ' is negative.']);
9   end
10
11  if num == 0
12      disp([num2str(num) ' is zero.']);
13  end
```

Command Window

```
>> if26
Enter a number: 5
5 is positive.
>>
```



Conditional statements: If Else

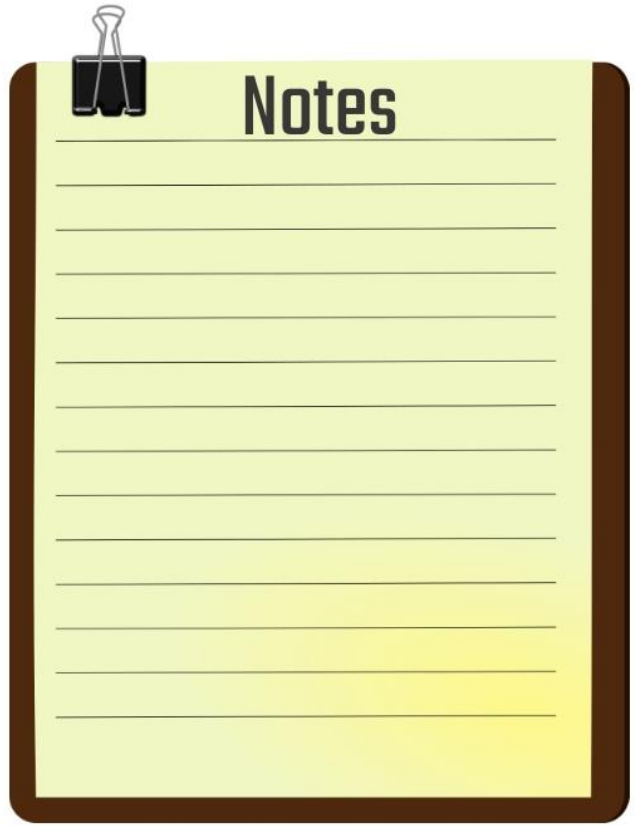
In conditional if Statement the additional block of code is merged as else statement which is performed when if the condition is false else condition won't be executed when if the condition is True.

Editor - C:\Users\DELL\Documents\MATLAB\els6.m

els6.m

```
1  x = 3;
2  if x > 5
3      disp('x is greater than 5');
4  else
5      disp('x is less than or equal to 5');
6  end
```

```
x is less than or equal to 5
>>
```



Conditional statements: If Else

Write a program in Matlab that determines whether the entered character is the letter A or not

Editor - C:\Users\DELL\Documents\MATLAB\a_or_no.m

a_or_no.m

```
1  input_char = input('Enter a character: ', 's');
2  if input_char == 'A' || input_char == 'a'
3      disp('The entered character is A');
4  else
5      disp('The entered character is not A');
6  end
```

Command Window

```
>> a_or_no
Enter a character: t
The entered character is not A
fx >>
```

Conditional statements: If-elseif-else

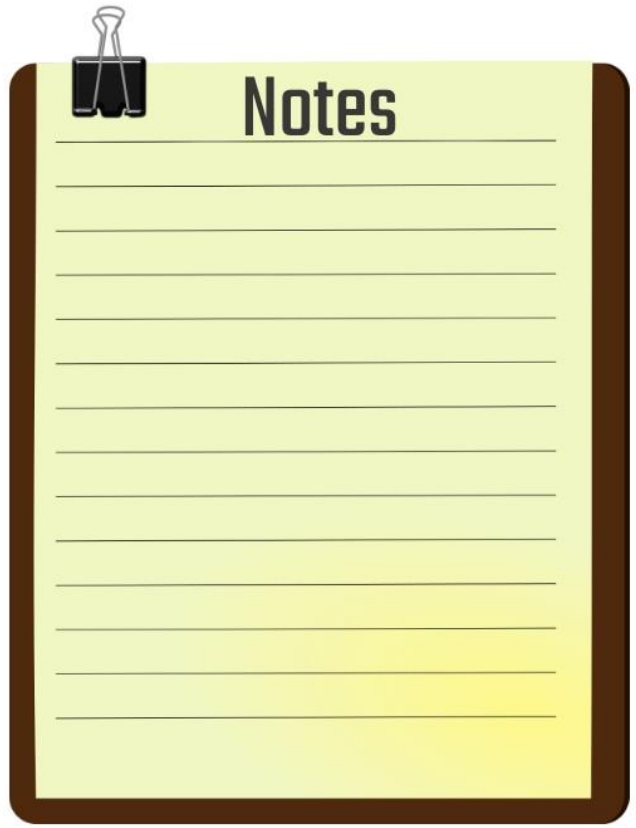
- An if statement can be followed by one (or more) optional elseif and an else statement, which is very useful to test various conditions.

Editor - C:\Users\DELL\Documents\MATLAB\if_else_ifelse.m

if_else_ifelse.m

```
1   x = 10;
2   if x > 10
3       disp('x is greater than 10');
4   elseif x < 10
5       disp('x is less than 10');
6   else
7       disp('x is equal to 10');
8   end
```

```
x is equal to 10
>>
```



Conditional statements: If-elseif-else

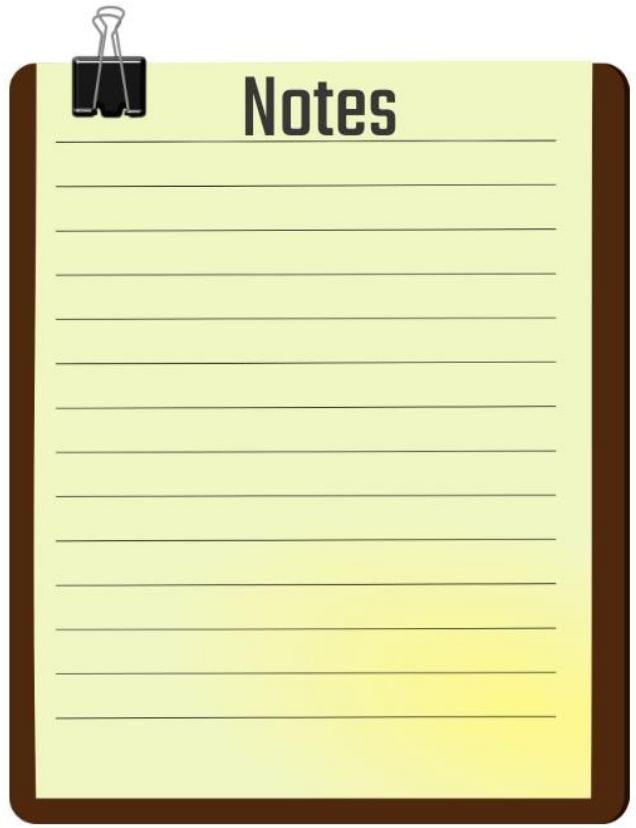
Write a MATLAB program that classifies a number as positive, negative, or zero using only if statements.

ifelse6.m

```
1  num = input('Enter a number: ');
2
3  if num > 0
4      disp('The number is positive.');
5  elseif num < 0
6      disp('The number is negative.');
7  else
8      disp('The number is zero.');
9  end
```

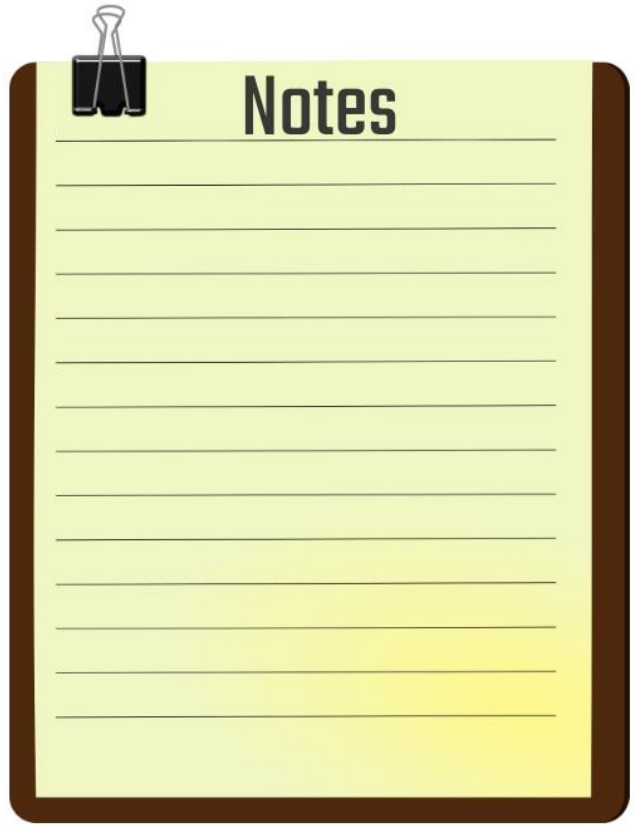
Command Window

```
>> ifelse6
Enter a number: 2
The number is positive.
>>
```



Conditional statements:

Write a MATLAB program that checks if a given number is odd, even, or zero.



Conditional statements: Switch-case

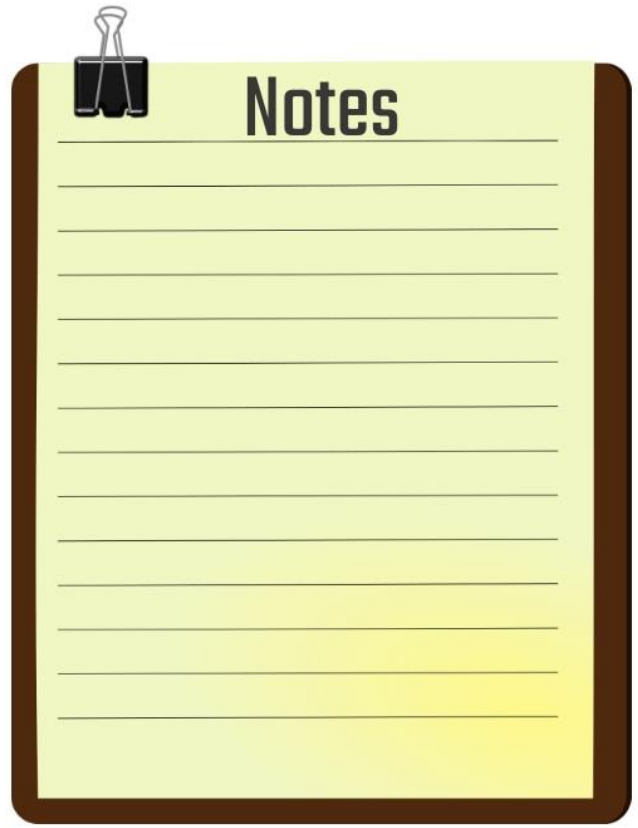
- A switch block is familiar to if-elif-else-end statements. It works as same as in other languages except for syntax. But there are few key differences between them. A switch block conditionally executes one set of statements from several choices. Each choice is covered by a case statement.

Editor - C:\Users\DELL\Documents\MATLAB\switch_.m

switch_.m

```
1   day = 'Monday';
2   switch day
3       case 'Monday'
4           disp('Today is Monday');
5       case 'Tuesday'
6           disp('Today is Tuesday');
7       otherwise
8           disp('Today is neither Monday nor Tuesday');
9   end
```

```
Today is Monday
>>
```



Questions

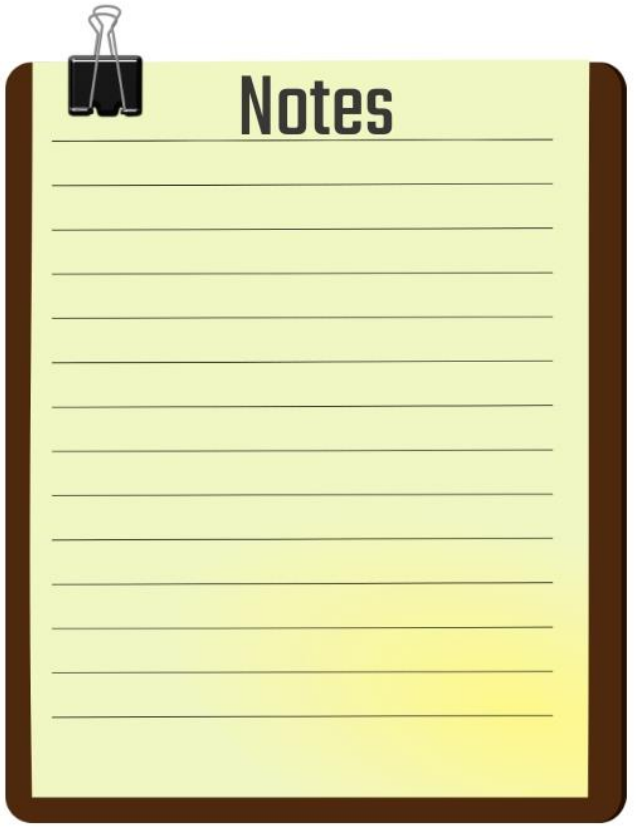
- 1-Write a MATLAB program that classifies a student's grade based on their score.
- 2-Write a MATLAB program that checks if a given number is positive.
- 3-Write a MATLAB program that determines the day of the week based on a user-inputted number (1 for Monday, 2 for Tuesday, etc.).
- 4-Give me a programming example for each:

IF

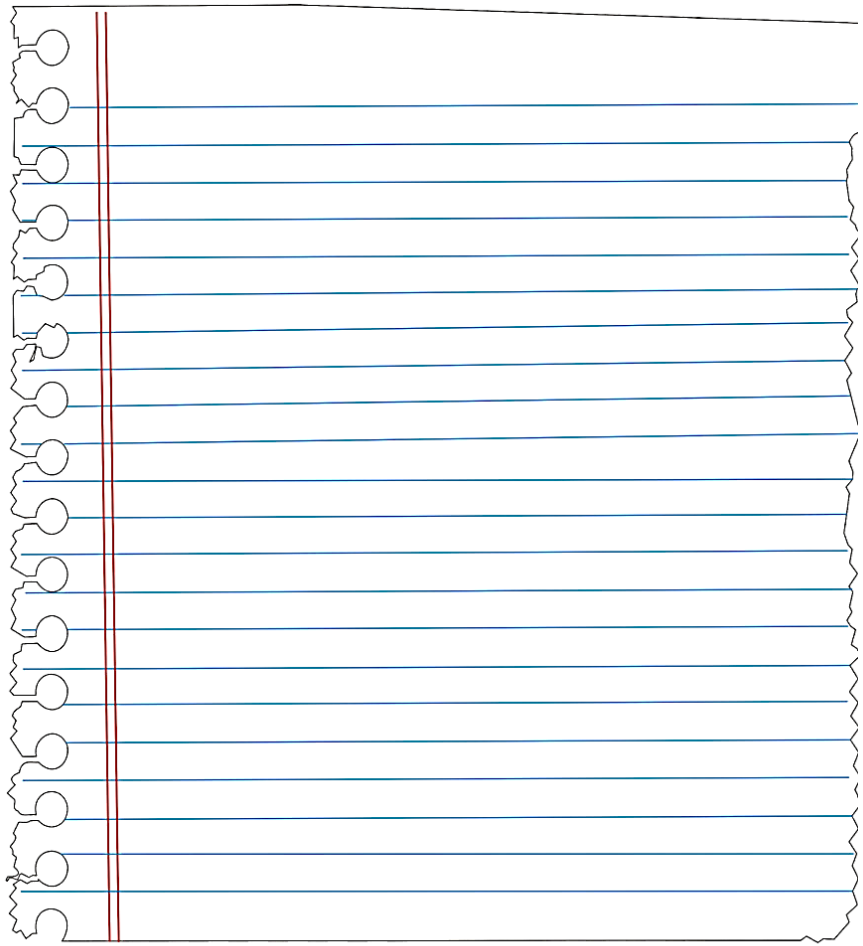
If_Else

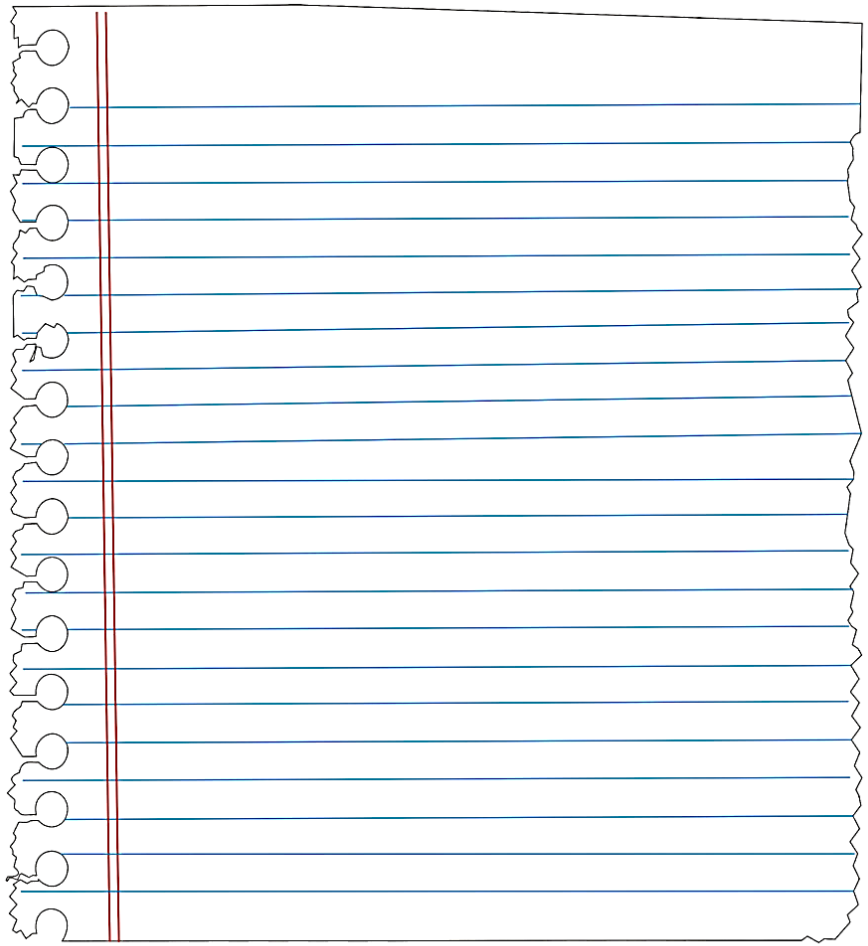
If_elseIf_Else

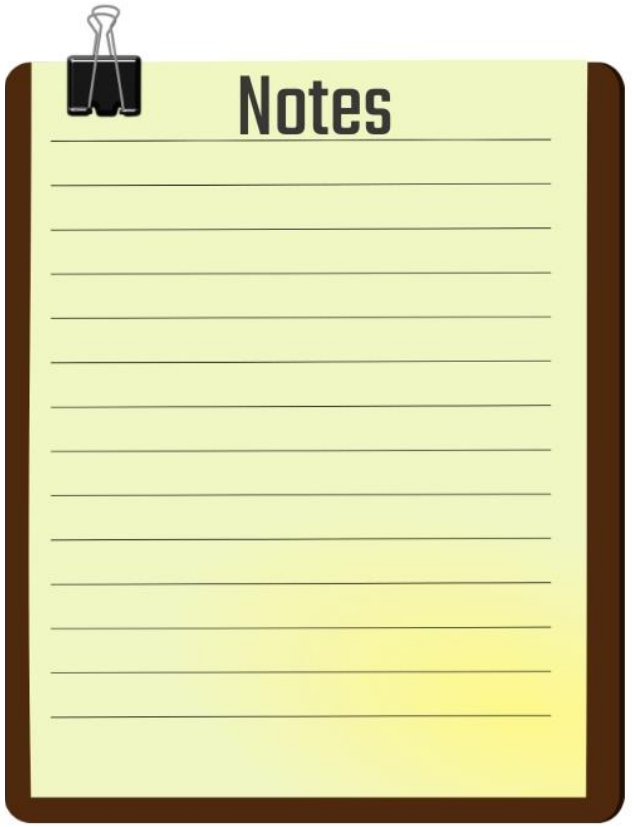
Switch_case



Solutions







Computer programming and applications MATLAB-beginner

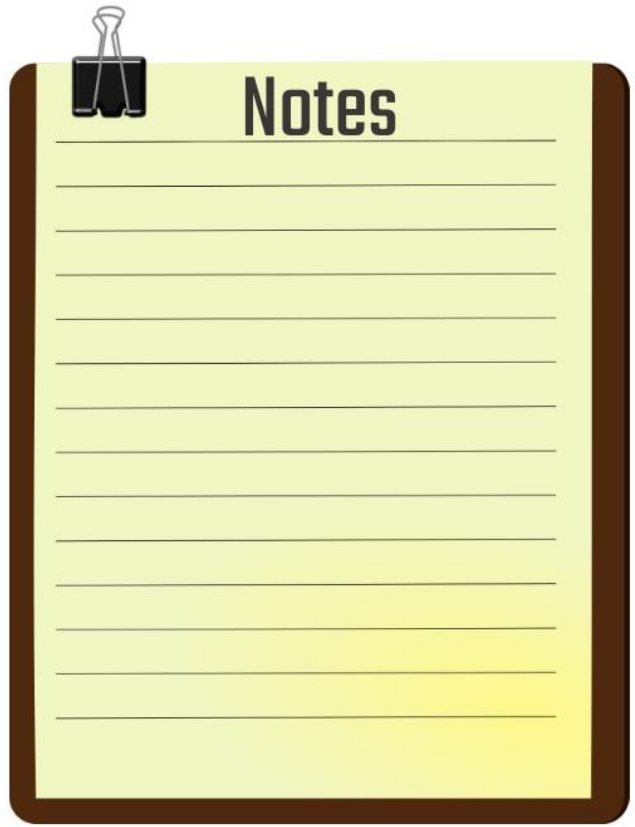
Repetition statementsents

Assist.Lec Aya Abdel Hussein

Repetition statementsents

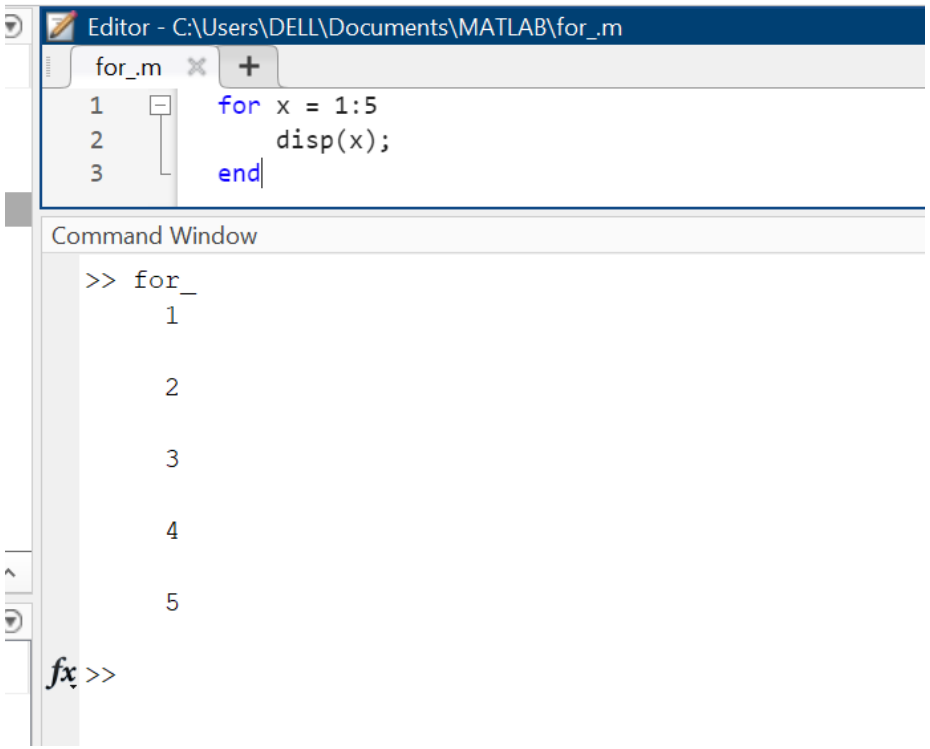
- Repetition statements, also known as loops, are programming constructs that allow you to repeat a block of code multiple times. They are used when you need to perform a task iteratively, such as processing elements in an array, iterating over a sequence of numbers, or executing a block of code until a certain condition is met.
- In MATLAB, there are mainly two types of repetition statements:

- 1- for loop: The for loop iterates over a sequence of values and executes a block of code for each value in the sequence.
- 2- while loop: The while loop repeats a block of code as long as a specified condition is true.

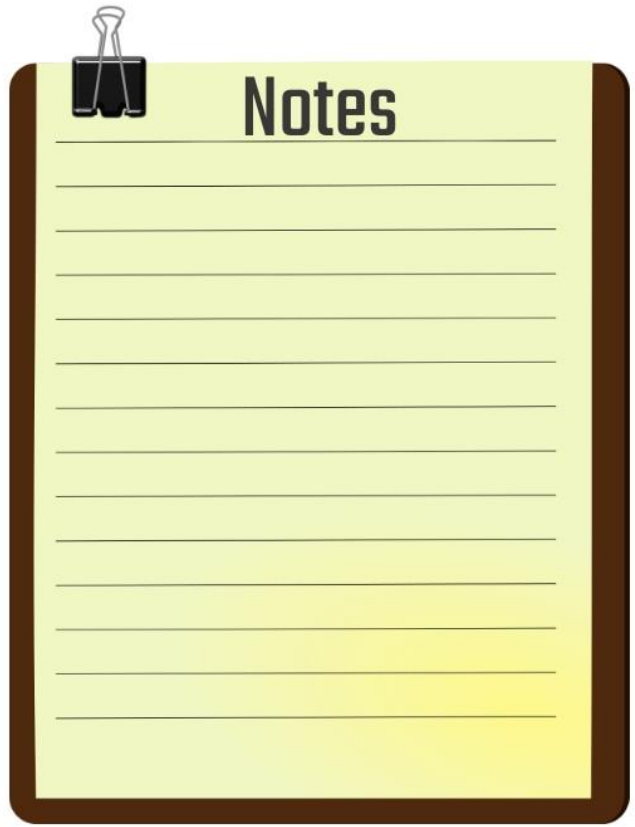


for Statement: Example of printing numbers

- The following example is to print numbers from 1 to 5 using for statement
- using a for statement in MATLAB does shorten and simplify the process of writing repetitive print instructions or any repetitive task.




```
disp(1);
disp(2);
disp(3);
disp(4);
disp(5);
```



for Statement: Sum of Numbers Example

- Explanation:
- total is initialized to zero.
- The for loop iterates over integers from 1 to 10.
- In each iteration, the current value of i is added to total.
- After the loop, the sum of the numbers from 1 to 10 is displayed.

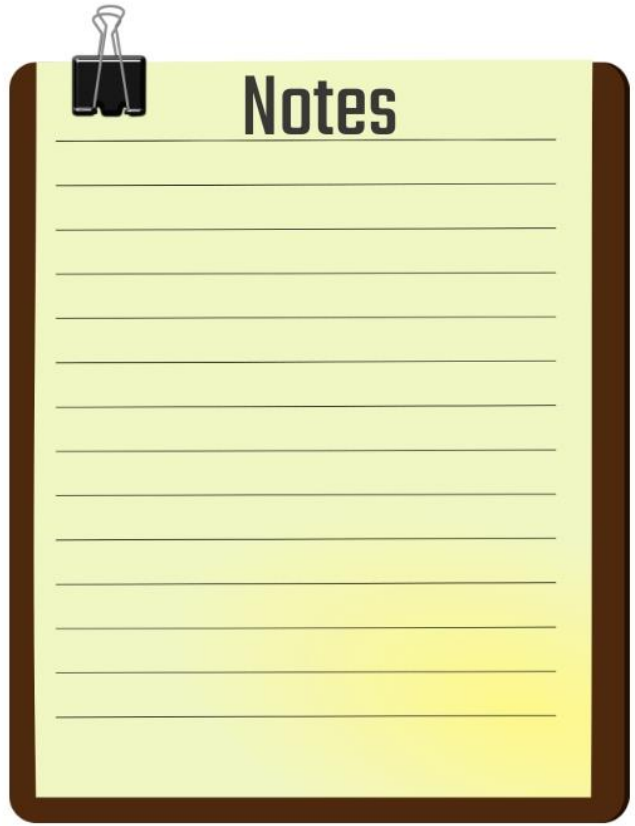
Editor - C:\Users\DELL\Documents\MATLAB\sum10int.m

sum10int.m

```
1    total = 0;  % Initialize the variable to store the sum
2    for i = 1:10
3        total = total + i;
4    end
5    disp(['The sum of the first 10 integers is ', num2str(total)]);
```

Command Window

```
>> sum10int
The sum of the first 10 integers is 55
>>
```



for Statement:

- Write a MATLAB program to separate elements in the given array into two arrays - one containing elements larger than ten and the other containing elements smaller than ten.

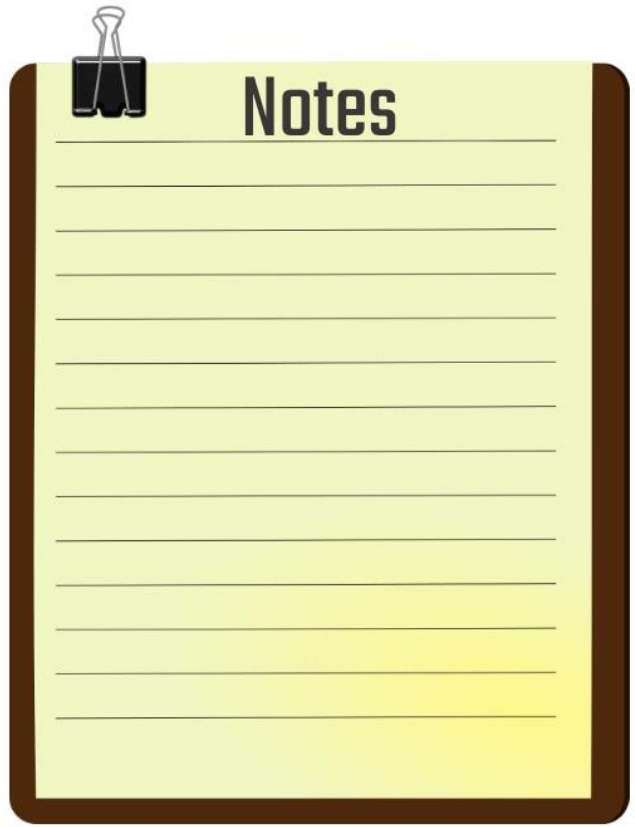
Editor - C:\Users\DELL\Documents\MATLAB\larger_than_ten.m

larger_than_ten.m

```
1   array = [3, 12, 5, 8, 15, 10, 20, 7, 25];
2   larger_than_ten_ = [];
3   smaller_than_ten = [];
4   for i = 1:length(array)
5       if array(i) > 10
6           larger_than_ten_ = [larger_than_ten_, array(i)];
7       elseif array(i) < 10
8           smaller_than_ten = [smaller_than_ten, array(i)];
9       end
10  end
11  disp('Elements larger than ten:');
12  disp(larger_than_ten_);
13  disp('Elements smaller than ten:');
14  disp(smaller_than_ten);
```

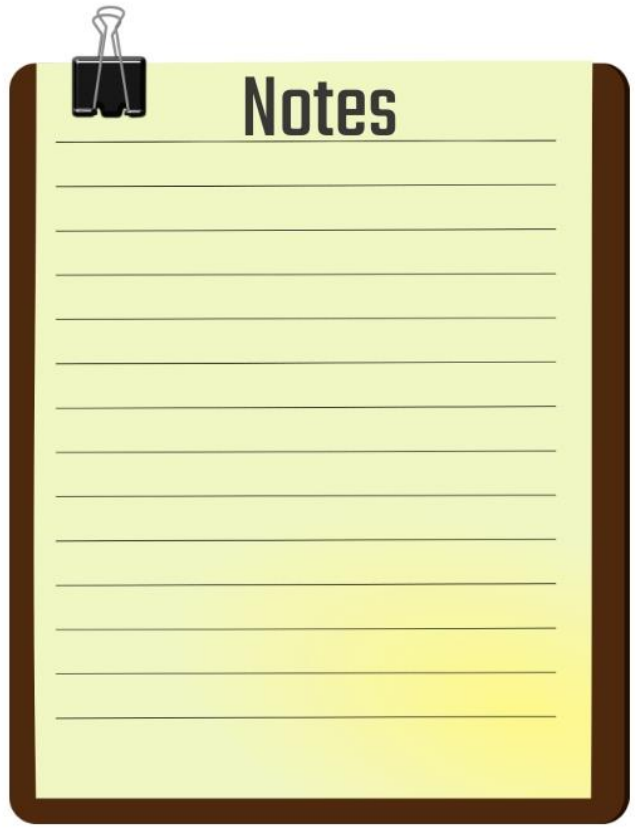
Command Window

```
>> larger_than_ten
Elements larger than ten:
    12    15    20    25

Elements smaller than ten:
     3     5     8     7
```



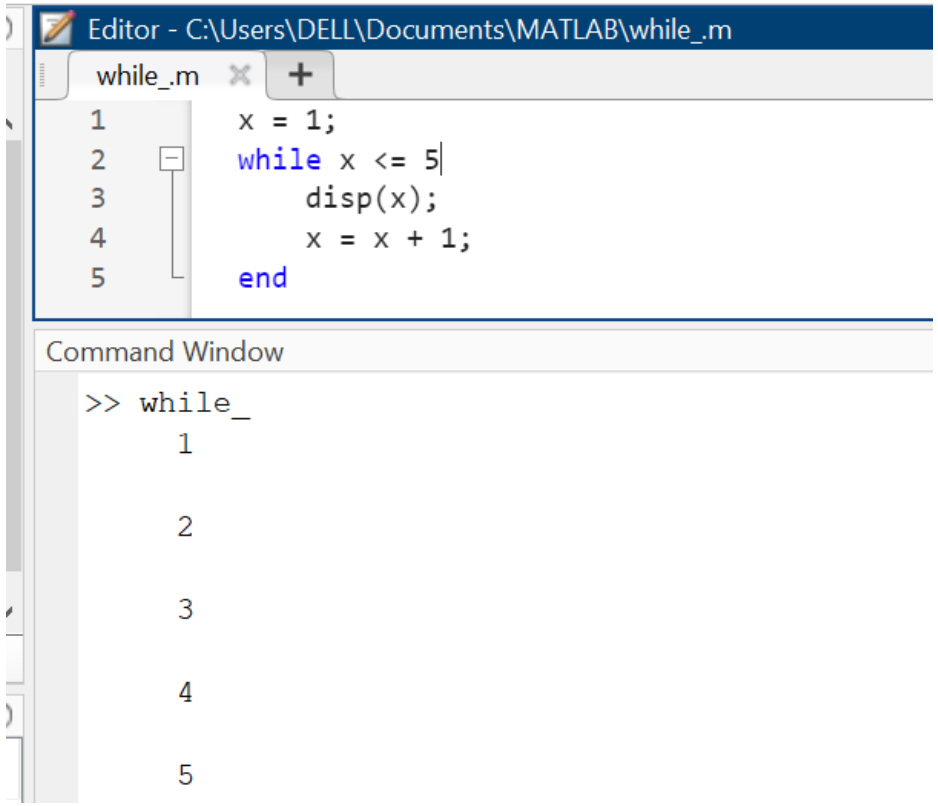
for Statement:

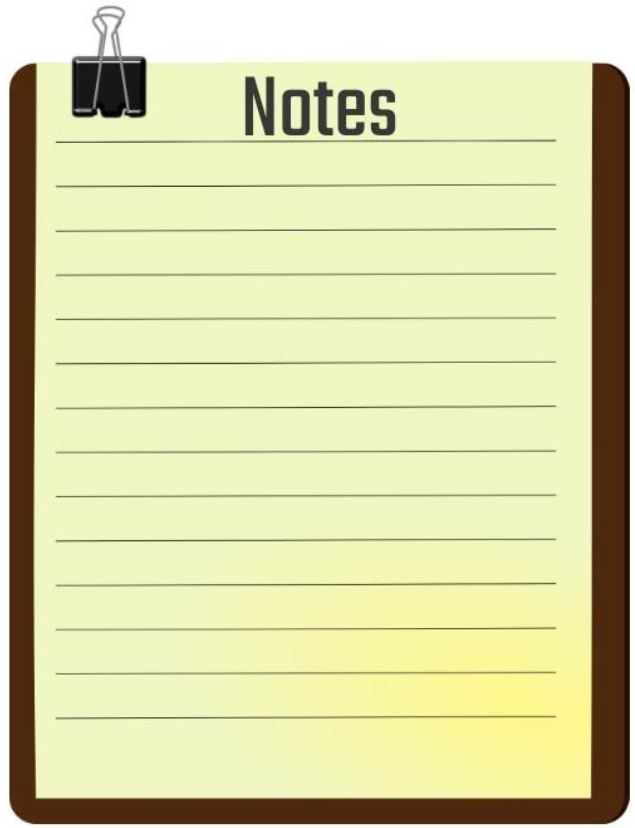
- Write a MATLAB program that prompts the user to enter a number and then prints the number repeatedly, with the number of repetitions equal to the value of the entered number.



while Statement:

- while loop: The while loop repeats a block of code as long as a specified condition is true.
- This loop will print the numbers 1 to 5.





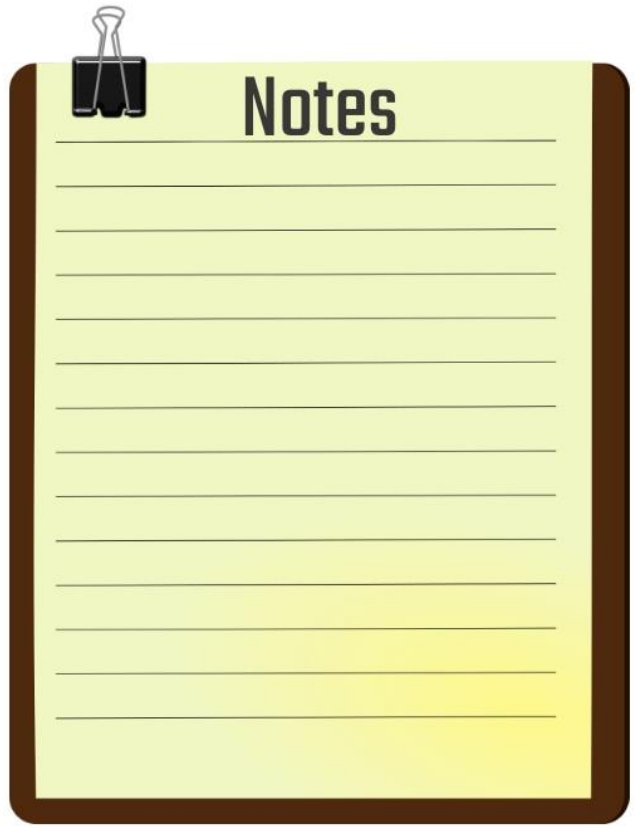
while Statement:

- Write a program that prints numbers from 1 to 10 using a while loop.

```
i = 1;
while i <= 10
    disp(i);
    i = i + 1;
end
```

Write a program that counts down from 10 to 1 using a while loop.

```
i = 10;
while i >= 1
    disp(i);
    i = i - 1;
end
```

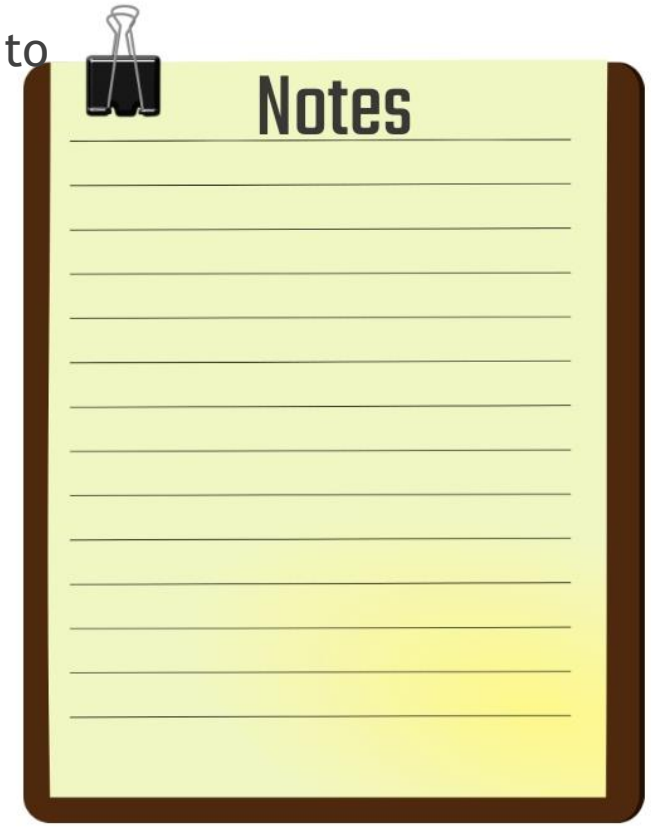


Questions

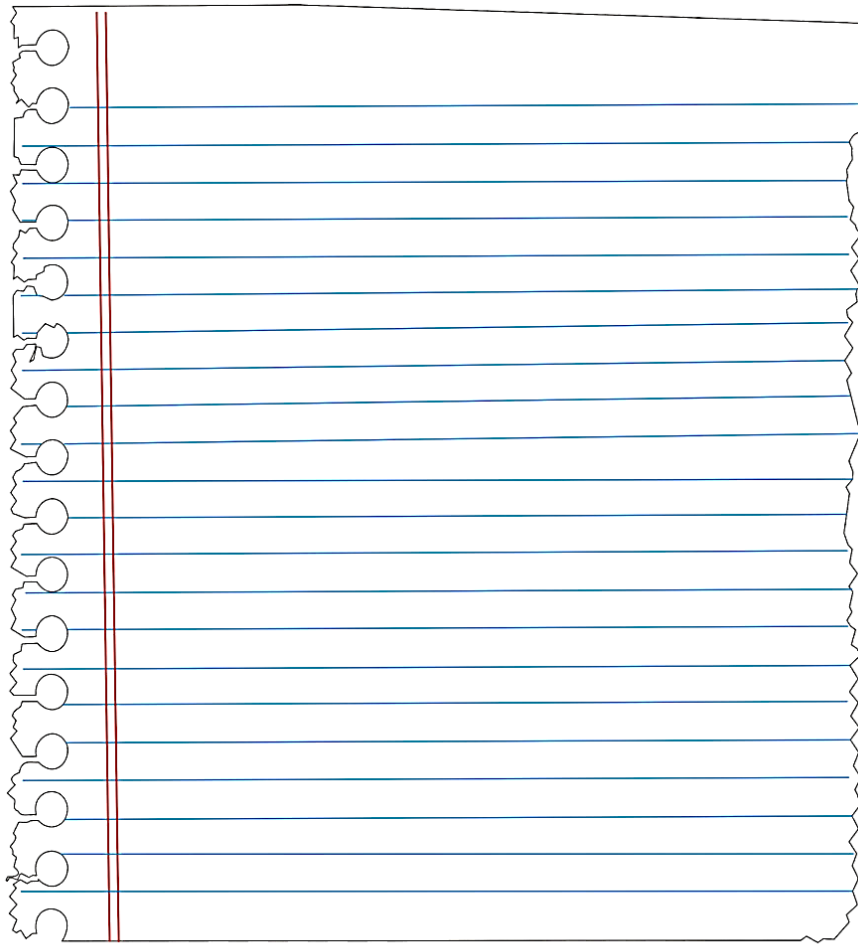
- 1- Write a program in Matlab that reads the student's name and grades for seven subjects and then prints the student's name and grade point average

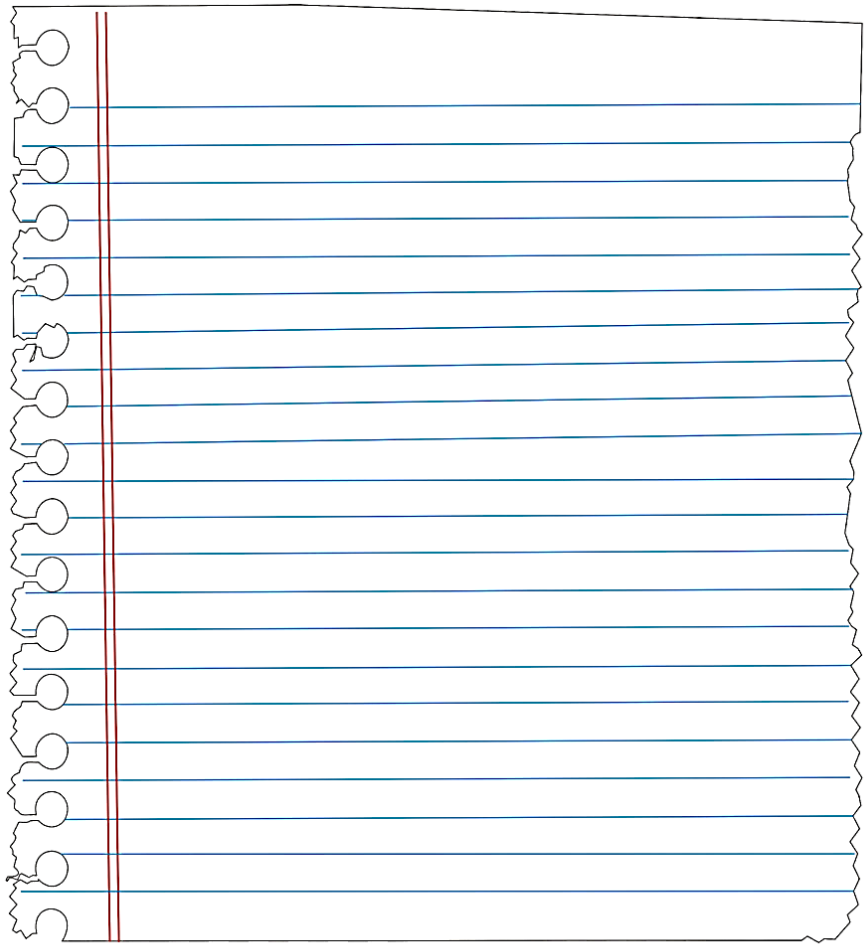
Note: Use for

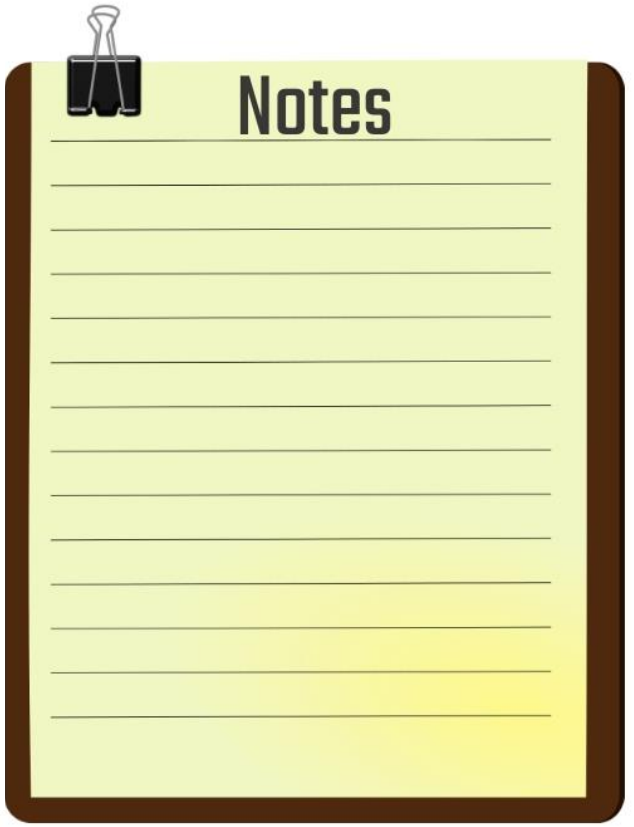
- 2- Write a program in Matlab to print numbers in descending order from 10 to 1, once using for and once using while.
- 3- what is Repetition statements
- 4- define for loop
- 5- define while loop



Solutions







Computer programming and applications MATLAB-beginner

Combination of conditional and repetition statements

Assist.Lec Aya Abdel Hussein

Combination of conditional and repetition statements

- Combining conditional and repetition statements in MATLAB can be very powerful for implementing complex algorithms and solving various computational problems efficiently.
- Given a loop in MATLAB that iterates from 1 to 10, write the code snippet that prints a message based on the value of the loop index i:

If i is less than 5, the message "Less than 5" is displayed.

If i is equal to 5, the message "Equal to 5" is displayed.

If i is greater than 5, the message "Greater than 5" is displayed.

The code should execute the loop and print the appropriate message for each value of i.

for_if.m

```
for i = 1:10
    if i < 5
        disp('Less than 5');
    elseif i == 5
        disp('Equal to 5');
    else
        disp('Greater than 5');
    end
end
```

Command Window

```
>> for_if
Less than 5
Less than 5
Less than 5
Less than 5
Equal to 5
Greater than 5
Greater than 5
Greater than 5
Greater than 5
Greater than 5
```

Combination of conditional and repetition statements

- Given a loop in MATLAB that iterates from 1 to 10, write the code snippet that prints a message based on the value of the loop index i:
- If i is less than 5, the message "Less than 5" is displayed.
- If i is equal to 5, the message "Equal to 5" is displayed.
- If i is greater than 5, the message "Greater than 5" is displayed.
- The code should execute the loop and print the appropriate message for each value of i.

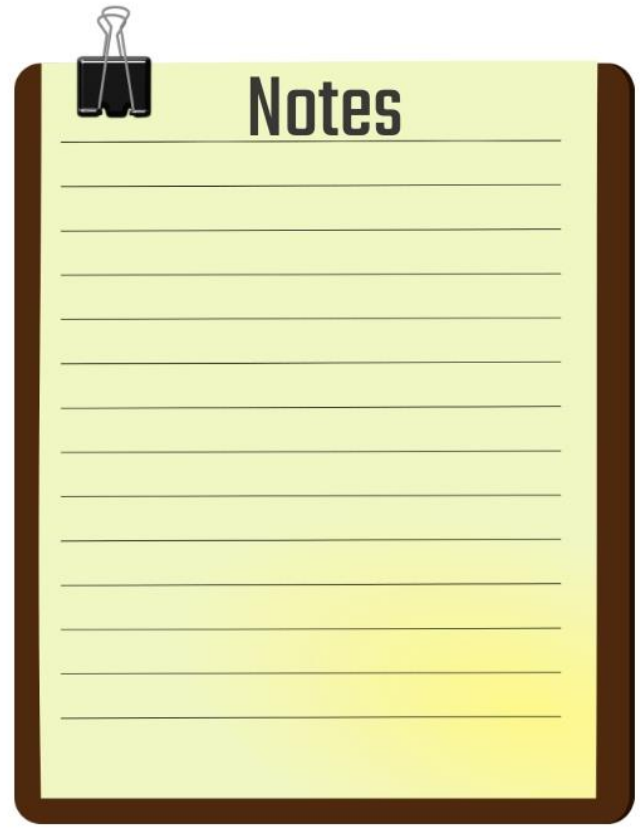
Editor - C:\Users\DELL\Documents\MATLAB\for_if2.m

for_if2.m

```
1   x = 0;
2   while x < 5
3       if x == 0
4           disp('Zero');
5       else
6           disp('Not zero');
7       end
8       x = x + 1;
9   end
```

Command Window

```
>> for_if2
Zero
Not zero
Not zero
Not zero
Not zero
>>
```



Combination of conditional and repetition statements

- Consider an array of numbers numbers containing the following values: [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]. Write a MATLAB code snippet that iterates through each number in the array, checks if the number is even, and if so, adds it to a new array called evens. After iterating through all numbers, the code should display the array evens, containing all the even numbers from the original array.

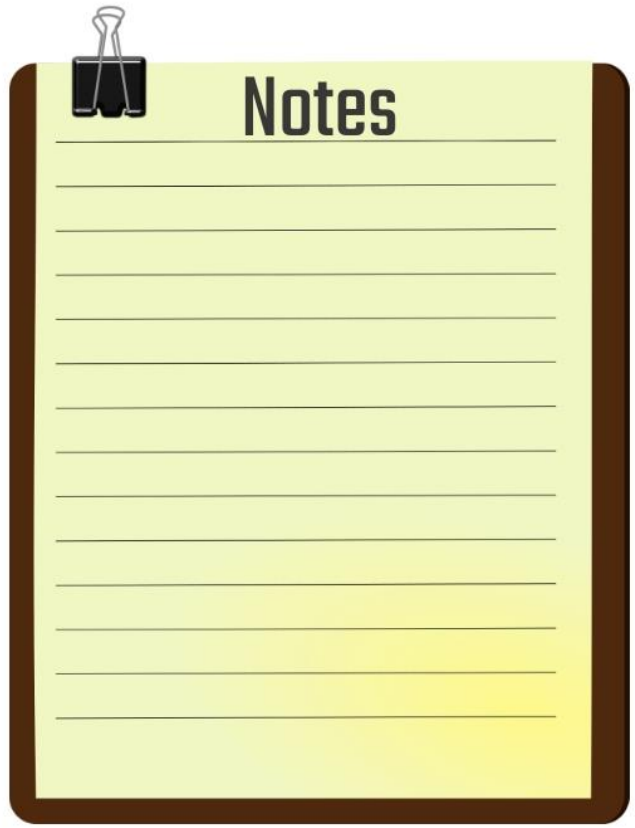
Editor - C:\Users\DELL\Documents\MATLAB\even.m

even.m

```
1  numbers = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10];
2  evens = [];
3  for num = numbers
4      if mod(num, 2) == 0
5          evens = [evens, num];
6      end
7  end
8  disp('Even numbers:');
9  disp(evens);
```

Command Window

```
>> even
Even numbers:
     2     4     6     8    10
```



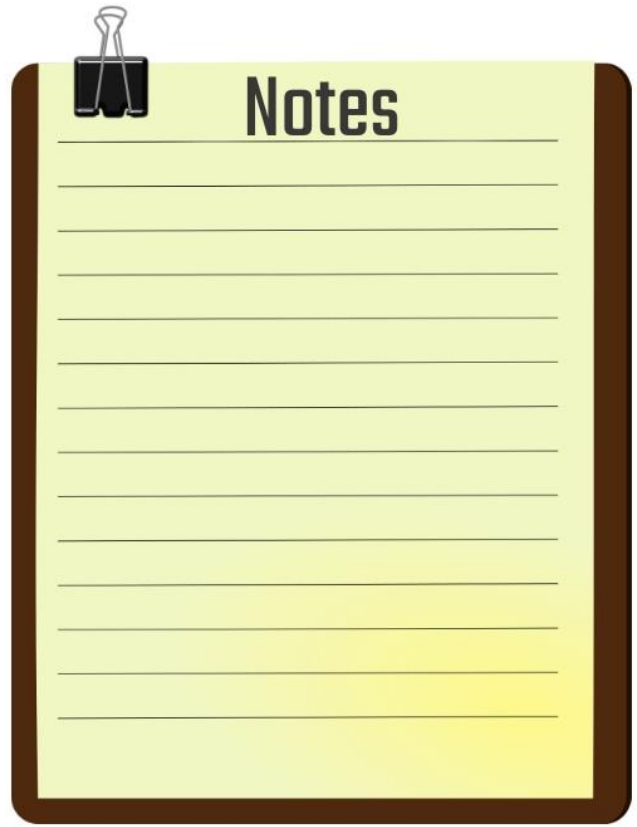
Combination of conditional and repetition statements

Given an array of student scores student_scores with the following values: [75, 85, 90, 60, 40, 55, 95, 80, 70, 65], write a MATLAB code snippet that iterates through each score, determines the corresponding grade based on the score using the following criteria:
A: score >= 90, B: score >= 80, C: score >= 70, D: score >= 60, F: score < 60
The code should output each score along with its corresponding grade

score.m

```
student_scores = [75, 85, 90, 60, 40, 55, 95, 80, 70, 65];
for i = 1:length(student_scores)
    s= student_scores(i);
    if s >= 90
        grade = 'A';
    elseif s >= 80
        grade = 'B';
    elseif s >= 70
        grade = 'C';
    elseif s >= 60
        grade = 'D';
    else
        grade = 'F';
    end
    fprintf('Score: %d, Grade: %s\n', s, grade);
end
```

```
Score: 75, Grade: C
Score: 85, Grade: B
Score: 90, Grade: A
Score: 60, Grade: D
Score: 40, Grade: F
Score: 55, Grade: F
Score: 95, Grade: A
Score: 80, Grade: B
Score: 70, Grade: C
Score: 65, Grade: D
```



Combination of conditional and repetition statements

- Write a program that examines an array consisting of ten numbers and determines the odd and even numbers for each number

Editor - C:\Users\DELL\Documents\MATLAB\oldeven.m

oldeven.m

```
1   numbers = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10];
2   for idx = 1:length(numbers)
3       num = numbers(idx);
4
5       if mod(num, 2) == 0
6           fprintf('%d is even\n', num);
7       else
8           fprintf('%d is odd\n', num);
9       end
10  end
```

Command Window

```
>> oldeven
1 is odd
2 is even
3 is odd
4 is even
5 is odd
6 is even
7 is odd
8 is even
9 is odd
10 is even
```

Computer programming and applications MATLAB-beginner

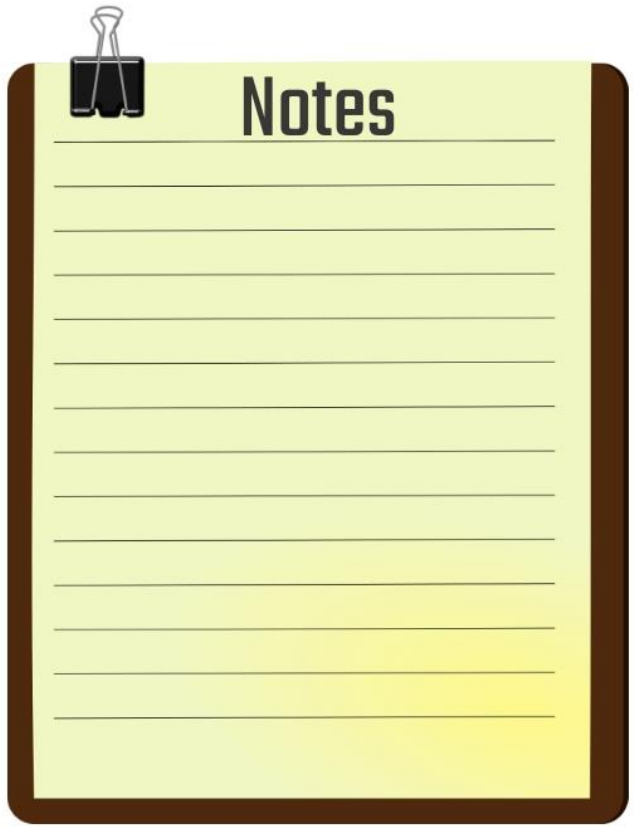
Basic Plotting

Assist.Lec Aya Abdel Hussein

Basic Plotting

- Basic Plotting: MATLAB has extensive facilities for displaying vectors and matrices as graphs, as well as annotating and printing these graphs.
- In MATLAB, the plot function is used to create 2D plots. It's a fundamental tool for visualizing data and mathematical functions. Here's a breakdown of its importance and usefulness with a simple example:

- 1-Data Visualization
- 2-Communicating Results
- 3-Exploratory Data Analysis
- 4-Debugging and Testing

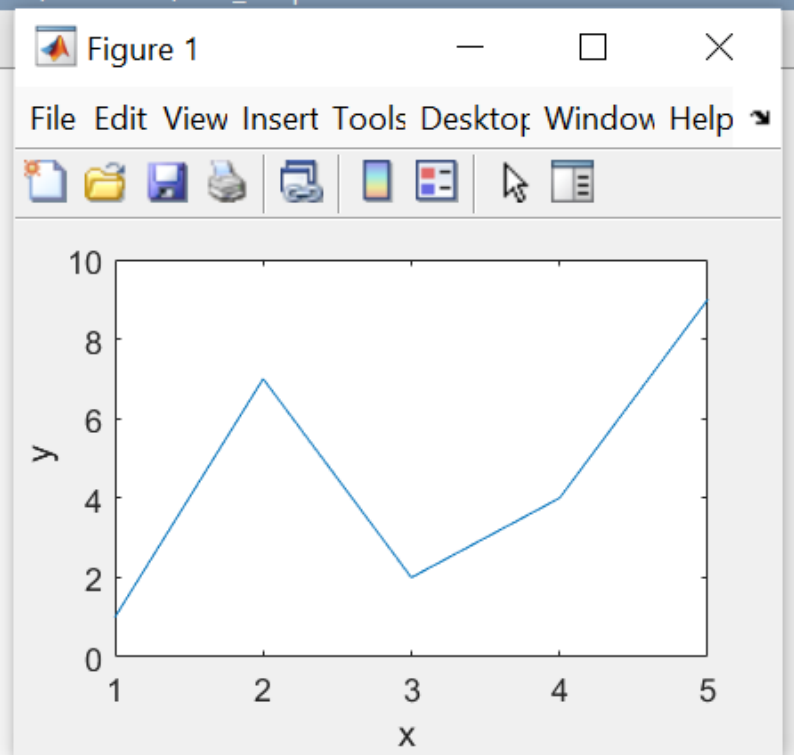


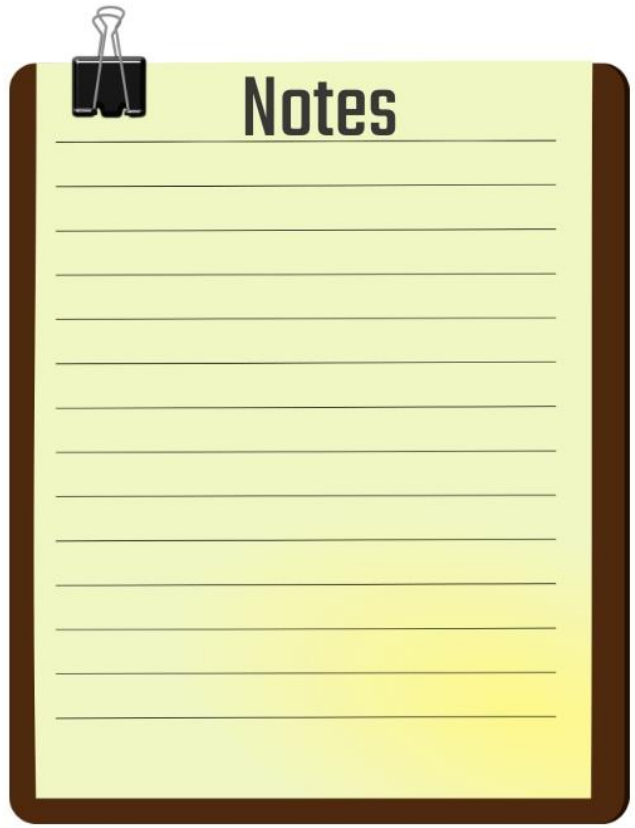
Basic Plotting

Editor - C:\Users\DELL\Documents\MATLAB\new_script.m

new_script.m

```
1   x=[1,2,3,4,5];
2   y=[1,7,2,4,9];
3   plot(x,y);
4   xlabel('x ');
5   ylabel(' y');
6
```

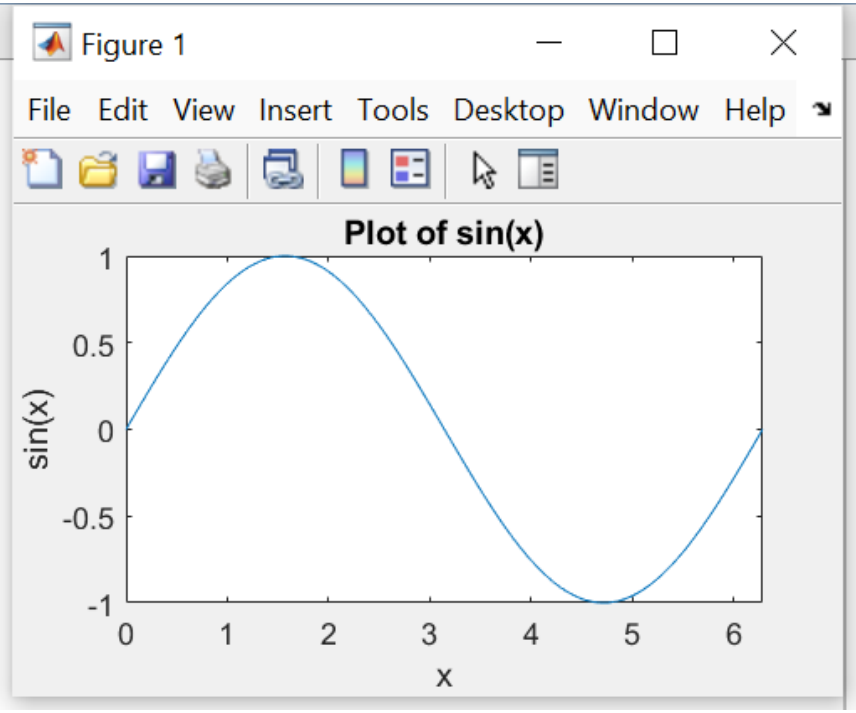




Basic Plotting

sin2.m

```
1  x = linspace(0, 2*pi, 2000);
2  y = sin(x);
3  plot(x, y);
4  xlabel('x');
5  ylabel('sin(x)');
6  title('Plot of sin(x)');
```



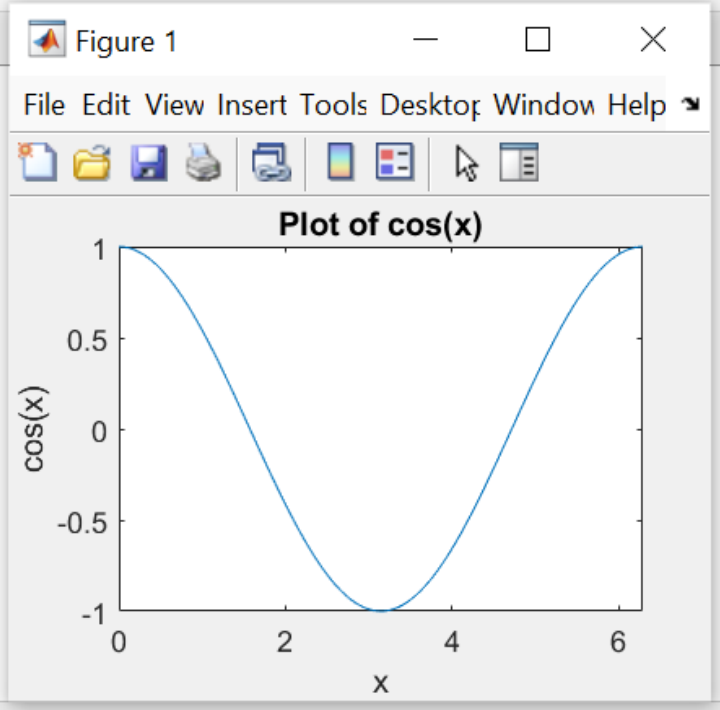
Notes

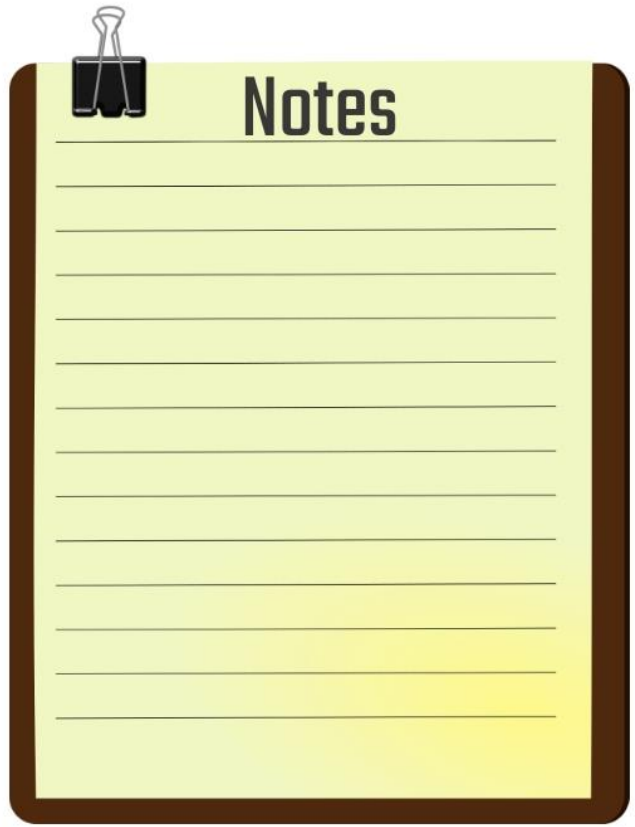
Basic Plotting

Editor - C:\Users\DELL\Documents\MATLAB\cos.m

cos.m

```
1   x = linspace(0, 2*pi, 100);
2   y = cos(x);
3   plot(x, y);
4   xlabel('x');
5   ylabel('cos(x)');
6   title('Plot of cos(x)');
```





Multiple Data Sets in One Graph

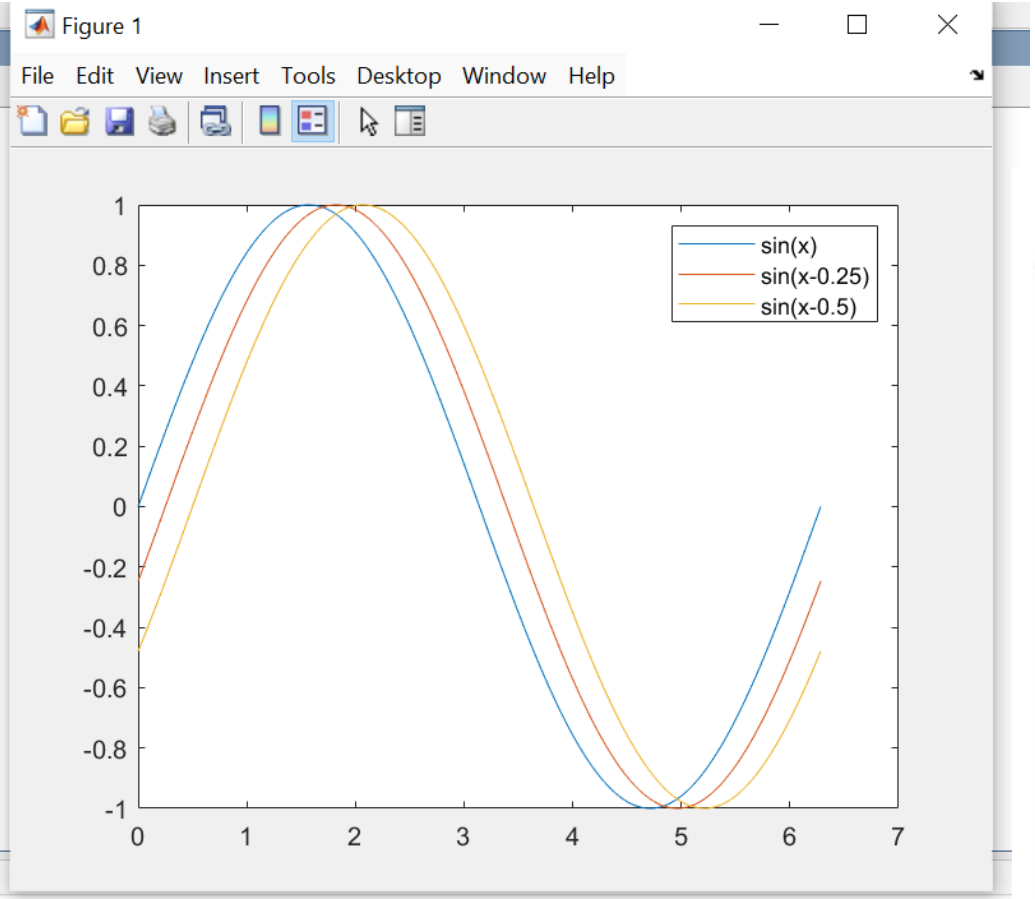
C: ▸ Users ▸ DELL ▸ Documents ▸ MATLAB

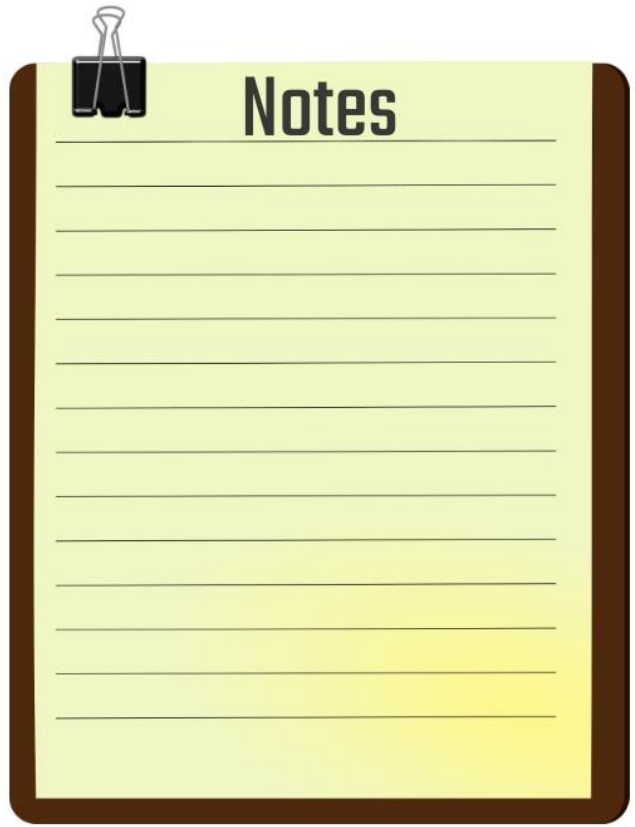
Users\DELL\Documents\MATLAB\multi.m

multi.m

```
x = 0:pi/100:2*pi;
y = sin(x);
y2 = sin(x - 0.25);
y3 = sin(x - 0.5);
plot(x, y, x, y2, x, y3);
legend('sin(x)', 'sin(x-0.25)', 'sin(x-0.5)');
```

ndow



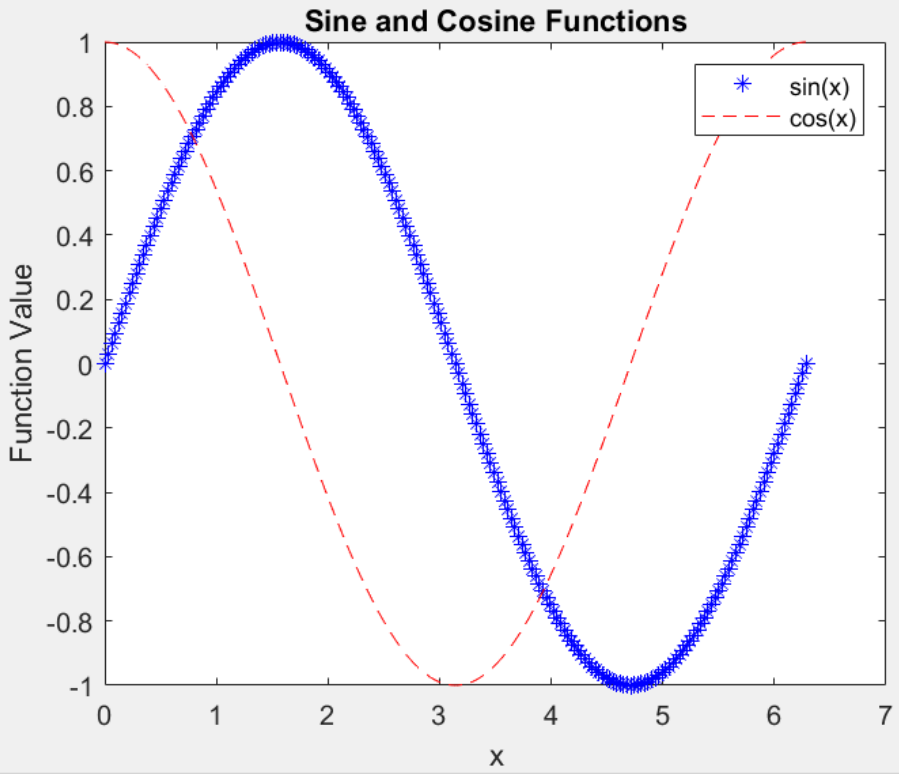


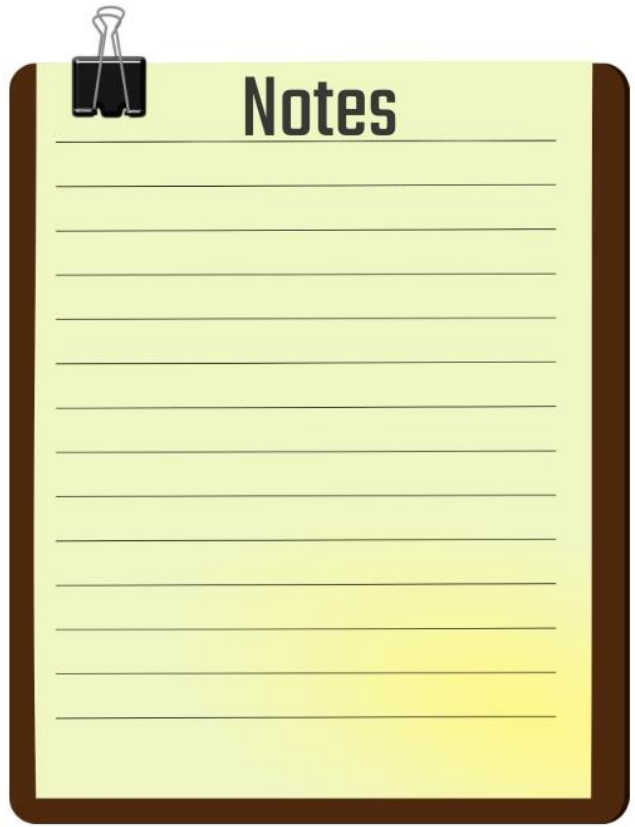
Specifying Line Styles and Colors

\Users\DELL\Documents\MATLAB\Specifying.m

g.m

```
x = 0:pi/100:2*pi;
y_sin = sin(x);
y_cos = cos(x);
plot(x, y_sin, 'b*', x, y_cos, 'r--');
xlabel('x');
ylabel('Function Value');
title('Sine and Cosine Functions');
legend('sin(x)', 'cos(x)');
```





Multiple Plots in One Figure

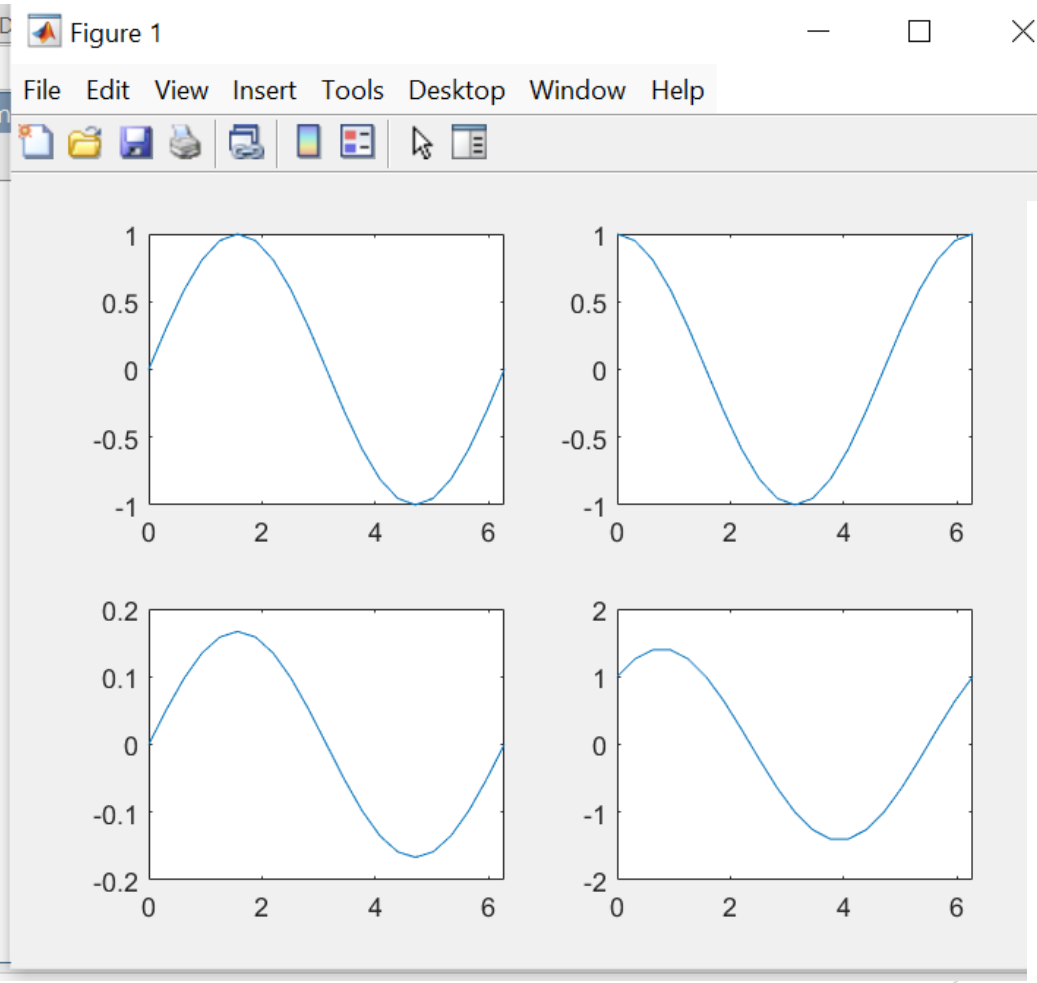
ILE NAVIGATE COD

C: ▸ Users ▸ DELL ▸ Documents ▸ MATLAB

Editor - C:\Users\DELL\Documents\MATLAB\multi_plots.m

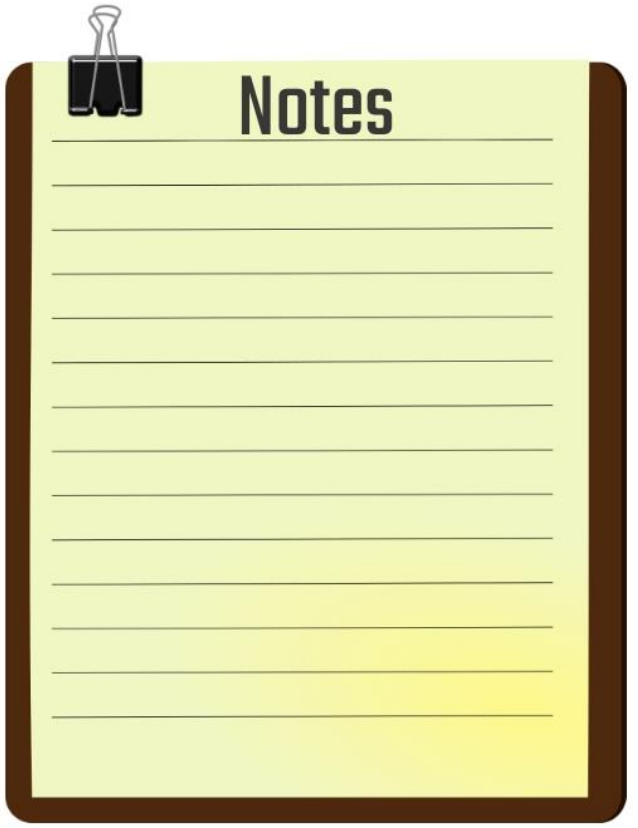
multi_plots.m

```
1   t = 0:pi/10:2*pi;
2   y1=sin(t);
3   y2=cos(t);
4   y3=sin(t)/6;
5   y4=cos(t)+sin(t);
6   subplot(2,2,1); plot(t,y1);
7   subplot(2,2,2); plot(t,y2);
8   subplot(2,2,3); plot(t,y3);
9   subplot(2,2,4); plot(t,y4);
10
```



Questions

- 1- Define basic plotting in MATLAB and explain its benefits.
- 2- Using basic plotting in MATLAB, create a plot of the sine function.
- 3- Using basic plotting in MATLAB, create a plot of the cos function.



Solutions