```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%  MATLAB Brush Up Course: Session 2  %%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%   by JOAN MARGALEF   %%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% LOGICAL/RELATIONAL OPERATORS, CONDITIONAL STATEMENTS, LOOPS & RAND. VARS

% This session focuses on logical and relational operators,
% conditional statements,  loops, and random variables in MATLAB.
% It starts with understanding basic logical operations and advances
% through if-else conditional statements for decision-making.
% Essential loop constructs such as 'for' and 'while' loops are discussed
% for iterative processes. The session also includes an introduction
% to generating and using random variables, emphasizing their role in
% simulations and stochastic processes.



%% 1. Logical and Relational Operators

% RELATIONAL OPERATORS
% == (equal to), ~= (not equal to), < (less than), > (greater than),
% <= (less than or equal to), >= (greater than or equal to).

% LOGICAL OPERATORS
% && (AND): True if both conditions are true.
% || (OR): True if at least one of the conditions is true.

% Note: & vs. &&. (same for | vs. ||)
% & element-wise logical AND operator: operates on arrays
% && short-circuit logical AND operator: operates on logical expressions


% Applying them to Objects
% Define a vector
vec = [1, 2, 3, 4, 5];
mat = [1 1 ; 4 5];

% Find elements greater than 3
greaterThanThree = vec > 3
smallerThanfour = mat < 4


% Using Logical Operations to Reference Matrix Elements

% Define a matrix
A = [-1, 2, -3; 4, 5, -6];

% Creates a Vector with elements meeting the condition
A(A < 0)

% Replace negative elements in A with 1
A(A < 0) = 1;
A

%Replace positve elemnets of first row by 5
A(1, A(1, :) > 0) = 5;
A

% Find elements in A that are greater than 2 and less than or equal to 5
selectedElements = A(A > 2 & A <= 5);
selectedElements
```

```
%% Practice 1

% Use the matrix: [1, 6, 3; 9, 2, 7; 4, 8, 5] for this exercise.

% Given a matrix, perform the following operations:
% 1. Replace all elements in the matrix that are greater than 5 with 10.

% 2. Find and display all elements that are less than or equal to 2
%    or greater than 8.
```

```matlab
%% 2.1. Conditional Statements: 'if', 'elseif', 'else'

% Conditional statements allow for executing different blocks of code based
% on specified conditions.

% - 'if': if the condition is true, the code block inside is executed.
% - 'else': used along with 'if', when the 'if' condition is false, the
%           code block inside is executed
% - 'elseif': used along with 'if', it covers other cases defined by other
%             conditions. When the 'elseif' condition is true, the
%             code block inside is executed


% Example:

number = 5;
if number > 0
    disp('The number is positive.');
end      % you need the end always!

% Adding 'else' to the 'if' statement
number = -4
if number > 0
    disp('The number is positive.');
else
    disp('The number is not positive.');
end

% Using 'elseif' for multiple conditions
number=0
if number > 0
    disp('The number is positive.');
elseif number < 0
    disp('The number is negative.');
else
    disp('The number is zero.');
end


%% Practice 2.1.

% Write a script using conditional statements to classify student's grade.
% The script should take a numerical grade (0-100) as input and output the
% corresponding letter grade based on the following scale:
% A: 90 and above
% B: 80 to 89
% C: 70 to 79
% D: 60 to 69
% F: Below 60





% NOTE: In MATLAB, the hierarchy of if, elseif, and else statements works
% in a top-down approach, evaluating each condition in the order they are
% written until one is found to be true.
```

```matlab
%% 2.2. Conditional Statements: Joint Conditions

% Joint conditions in MATLAB use logical operators 'AND' (&&) and 'OR' (||)
% to combine multiple criteria in a conditional statement.

% Example: Check if a number is greater than 0 and less than 10
number = 5;
if number > 0 && number < 10 % Both conditions must be true
    disp('Number is greater than 0 and less than 10.');
else
    disp('Number does not meet the conditions.');
end

% Example using 'OR' (||)
% Check if a number is less than 0 or greater than 10
if number < 0 || number > 10 % At least one condition must be true
    disp('Number is less than 0 or greater than 10.');
else
    disp('Number is between 0 and 10, inclusive.');
end

% Example using 'AND' and 'OR' together
% Check if a grade is passing and either in the A or B range
grade_final = 85;
grade_ex1=50;
grade_ex2=60;
eval=1;   %1 means unique, 0 continuous
if (eval == 1 && grade_final >= 50) || ...   % ... line break of the code
        (eval == 0 && grade_ex1 >= 60 && grade_ex1 >= 60)
    disp('Pass');
else
    disp('Do not pass');
end

% Note: Use parentheses to group conditions and order evaluation,
% especially when combining '&&' and '||' in the same statement.




%% Practice 2.2.

% Determine if an individual is eligible for a subsidy based on multiple
% criteria.

% The criteria are as follows:
% 1. Have an annual income less than $30,000, OR be a student.
                        % AND
% 2. Have at least two dependents OR be over the age of 65.

% Assume variables for annual income, student status (1 for student,
% 0 for non-student), number of dependents, and age are given.

% Use logical operators 'AND' (&&) and 'OR' (||).
% The script should output whether the individual is eligible for the
% subsidy or not.
% Test your script with different scenarios to ensure it works correctly.
```

```matlab
%% 3. Loops

clc;clear

% There are two kind of loops: 'for' and 'while'

% - 'for': run commands FOR a set of values

% - 'while': run commands WHILE something is true


%% 3.1. Loops: 'for' loops


% A 'for' loop is used to repeat a group of statements a fixed
% predetermined number of times.

% Example: Print numbers from 1 to 5
for i = 1:5    % 'i' is the var that will take each value at each iteration
    disp(i)
end            % you need the end always!


% Another example with vectors
x=zeros(1,10);
for i=1:10
    x(1,i)= 10 + i;
end
x


%% Practice 3.1.

% 1. Create a vector that stores the first 10 even numbers.
% Then, compute the sum of these numbers and display the result.

% 2. Create a matrix with two columns and 10 rows.
% In the first column, store the first 10 numbers of the Fibonacci series.
% In the second column, store the cumulative sum of the Fibonacci series up
% to that point.
```

```matlab
%% 3.2. Loops: Nested Loops


%  Nested loops in MATLAB allow the execution of a set of statements
%  multiple times, with each set of iterations performed within another
%  loop. (Loops over Loops)

% Example: Printing Combinations of Names and Surnames

% List of names and surnames (LIST have CELLS)
names = {'Alice', 'Bob', 'Charlie'};
surnames = {'Smith', 'Johnson', 'Williams'};

% You acces them with {}
names{1}

% Nested loop to print all combinations of names and surnames
disp('List of Names and Surnames:');
for i = 1:length(names)
    for j = 1:length(surnames)
        combinedName = [names{i} ' ' surnames{j}];
        disp(combinedName);
    end
end


% Example: Storing Values in a Matrix

% Initialize a 3x3 matrix
n=3;
matrix = zeros(n, n);

% Nested loop to fill the matrix with values
% Let's fill it with the product of its row and column indices
for i = 1:3
    for j = 1:3
        matrix(i, j) = i * j;
    end
end

% Display the resulting matrix
disp(matrix);


%% Practice 3.2.

% Given an array of initial GDP values for different countries and constant
% annual growth rate of 2%, calculate the GDP for each country over the
% next 'n' years.

% Store the GDP values for each year in a matrix, where each row represents
% a country and each column represents a year.

% Use 'n' as 10 for this exercise.
% Example initial GDPs: [1000, 2000, 3000] for three countries.
```

```
%% 3.3. Loops: While Loops

% A 'while' loop continues to execute a block of code as long as a
% specified condition is true.

% Example: Print numbers from 1 to 5
count = 1;
while count <= 5
    disp(count);
    count = count + 1;
end

%CAREFULL WITH WHILE LOOPS, YOU MIGHT CREATE AN INFINITE ONE!
count = 6;
while count ~= 5
    disp(count);
    count = count + 1;
end


% If we want a loop to stop before reaching its set, we can use Break
while count ~= 5
    disp(count);
    count = count + 1;

    if count==1000
        break
    end
end


% ECON EXAMPLE

% Initial investment amount
initialInvestment = 1000; % in dollars

% Annual interest rate
annualInterestRate = 0.05; % 5%

% Target amount
targetAmount = 1500; % The investment goal in dollars

% Initialize variables
currentAmount = initialInvestment;
years = 0;

while currentAmount < targetAmount

    currentAmount = currentAmount * (1 + annualInterestRate);

    % Increment the year count
    years = years + 1;

end

% Display the total number of years taken to reach the target amount
disp(['Total years taken to reach the target amount: ' num2str(years)]);
```

```
%% Practice 3.3.

% 1. Write a while loop to find the smallest integer 'n' such that
% the sum of the numbers from 1 to 'n' is greater than or equal to 100.
% Display the value of 'n'.


% 2. Given linear equations for demand and supply
% (demand = 100 - 2*price, supply = 2*price),
% use a while loop to find the equilibrium price (demand equals supply).
% Start with a price of 0 and increment in steps of 0.5 until equilibrium
% is reached.


% 3. Write a script that calculates the total cost of a product over time,
% where the cost increases by 5% each year due to inflation.
% The loop should terminate once the cost exceeds a budget limit.
% Start with an initial cost of $100 and a budget limit of $150.
```

```matlab
%% 4. Random Variables

% Working with random variables in MATLAB involves generating random
% numbers and simulating stochastic processes.



% Generating Basic Random Numbers

% Generate a single random number uniformly distributed (0, 1)
randNum = rand

% Generate a 3x3 matrix of random numbers uniformly distributed (0, 1)
randMatrix = rand(3, 3)

% If we want to keep the draw, use rng function to keep seed
rng(0); % Set the seed to a specific value (RUN TOGETHER)
randNumWithSeed1 = rand
randNumWithSeed2 = rand

% Standard Normal Distribution (mean=0, std=1)
randn(1,5); % 100 random numbers from standard normal distribution

% Normal Distribution (mean 'mu' and std 'sigma')
mu = 5; sigma = 2;
normrnd(mu,sigma,1,5)   % 1,5 are the vector dimension





% Simulating a Basic Random Walk

%       x_t = x_{t-1} + ε_t   (ε_t follows a Normal)

% Parameters for the random walk
numSteps = 100;     % Number of steps in the random walk
sigma=1;        % Standard deviation of the random step (ε_t)

% Initialize the array to store the values at each step
randomWalk = zeros(numSteps, 1);

% Start the random walk from an initial value, say 0
randomWalk(1) = 0;


% Generate the random walk
for t = 2:numSteps
    % Random step with normally distributed increments
    randomWalk(t) = randomWalk(t-1) + normrnd(0,sigma);
end



%% Practice 4

% Simulate an AR(1) process given by the equation:
% x_t = alpha * x_{t-1} + ε_t,
% where ε_t is a normally distributed random variable with mean 0 and
% standard deviation sigma.

% - Set alpha to a value different from 1 (e.g., 0.5)
% - Assume 100 steps for the process and a std of 1 for ε_t.
% - Initialize the first value of the process to 0
```