```matlab
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%  MATLAB Brush Up Course: Session 3  %%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%   by JOAN MARGALEF   %%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


% SYMBOLIC EXPRESSIONS, ANONYMOUS FUNCTIONS & LOCAL FUNCTIONS

% This session explores symbolic mathematics, anonymous functions and local
% functions. It covers symbolic expressions for computations in calculus
% and algebra, and anonymous functions for optimization. The concept
% of local functions is addressed for organizing complex scripts.



%% 0. Understanding Symbolic Expressions, Anonymous Funct., and Local Funct

% Symbolic Expressions:
% - Symbolic expressions involve symbolic math, where the values of
%   variables are not numerical but symbolic.
% - They are useful for exact mathematical calculations like
%   differentiation, integration, and solving equations symbolically.

% - Pros: Allow for precise manipulation of mathematical formulas.
% - Cons: Slower for numerical computations.
% - Purpose: Algebra

% Example:
syms x
symExpr = x^2 - 4*x + 4;

% Anonymous Functions:
% - Anonymous functions are quick, one-line functions.
% - They are ideal for simple operations.

% - Pros: Easy to write and use for simple tasks.
% - Cons: Less versatile for complex tasks.
% - Purpose: Function evaluation

% Example:
anonFunc = @(x) x^2 - 4*x + 4;

% Local Functions:
% - Help in organizing code and reusing functionality.

% - Pros: Useful for breaking down complex tasks within a script.
% - Cons: Not accessible outside the parent script or function.
% - Purpose: Reduce code

% This is an example, but will not work (see later why)
% Local function defined within a script or function file:
% function y = localFunc(x)
%     y = x^2 - 4*x + 4;
% end
```

```matlab
%% 1.1 Symbolic Expressions: Basics

% Initialize symbolic variables
syms x

% Basic Symbolic Operations
expr = x  - x^2 ;
expr


% Now we can do thins with the equations like

% - Differentiate w.r.t. x
diffExpr = diff(expr, x);
disp(diffExpr)

% Integrate the expression with respect to x
intExpr = int(expr, x);
intExpr

% Combine with other expressions
syms y z
expr2= y + z

expr3= expr + expr2

% And evaluate them:
evaluatedDiffExpr = subs(expr3, {x, y}, {1, 2})

%% Practice 1.1.

% 1. Define a symbolic expression f(x) = cos(x) - x^3.
%    Differentiate f(x) with respect to x, and then evaluate this
%    derivative at x = pi/4. Display the result.

% 2. Define a symbolic cost function C(x) = 50*x + 300
%    The revenue function R(x) is defined as R(x) = 75*x,
%    Find the profit function P(x) and compute the profit for producing
%    10 units.
```

```
%% 1.1 Symbolic Expressions: Solving Equations

clc;clear

syms x y z

% Define an equation
equation = x^2 - 4 == 0;
disp('Equation:');
disp(equation);

% Solve the equation
solution = solve(equation, x);
disp('Solution to the Equation:');
disp(solution);



% System of Equations

% Define a system of equations
eq1= x + y + z == 6
eq2= x - y + 2*z == 7
eq3= 2*x + y + z == 10

% Solve the system of equations as System
solutionsSystem = solve([eq1, eq2, eq3 ], [x, y, z]);
disp([solutionsSystem.x, solutionsSystem.y, solutionsSystem.z]);

% Solve the system of equations as indiviual sim vars
[xstar, ystar, zstar] = solve([eq1, eq2, eq3 ], [x, y, z]);

%% Practice 1.2.

% 1. Define and solve the equation x^3 - 3x^2 + 2 = 0 symbolically.
%    Display the solutions.

% 2. Define and solve the following system of equations symbolically:
%   2x + 3y - z = 1
%   -x + y + z = 3
%   x - 2y + 3z = -1
% Display the solutions for x, y, and z.
```

```matlab
%% 2.1. Anonymous Functions: Unconstrained Optimization

% Define an anonymous function for a quadratic equation
quadFunc = @(x) x - x^2 ;

% Easy to Evaluate
quadFunc(2)


% The advantage of being functions is MATLAB's set of built-in functions
% An important area where this is beneficial is in OPTIMIZATION.

% Optimization processes in MATLAB focus on finding LOCAL MINIMA!
% Local Minima are not Maximums! And also not necessarily Global minima!
% We will find ways to find them from searching local minima.

% As MATLAB performs numerical optimization, the results often depend on
% the chosen starting points, influencing the outcome of the minimization.

% There are several commands:

% - 'fminunc()' - Function Minimization Unconstrained (WE FOCUS ON THIS)
%     Finds a local minimum of a scalar function of one or more variables.
%     It is best suited for problems where the objective function is
%     differentiable and when gradient information is available or can be
%     approximated.
%
% - 'fminsearch()' - Unconstrained Multivariable Function Minimization
%     Uses the Nelder-Mead simplex algorithm to find a minimum of a function
%     of several variables. It does not require the function to be
%     differentiable, and is a good choice when gradient information is not
%     available.


quadFunc = @(x) x - x^2 ;
% This function has an inverse U-shape, implying it has two minima at
% negative and positive infinity. Depending on the starting point,
% the optimization process may converge to one of these minima.


% Use fminunc to find the minimum of the objective function
x0= 0 ;    % this is a guess, since optimize it numerically
fminunc(quadFunc2 , x0) % gives optimum

% If we also want the function evaluated at the optimium
[optimum, optimalValue] = fminunc(quadFunc , x0)


% With fminunc we can find also a maximum (add '-' to the function)
quadFunc2 = @(x)  - (x - x^2) ;
[optimum, optimalValue] = fminunc(quadFunc2 , x0)
maxValue = -optimalValue % Negate back to get the maximum utility

% Example objective function with two variables, x and y
%         f(x, y) = (x-1)^2 + (y-2)^2

% You define them as a vector now and call its separate elemenets
objectiveFunc = @(xy) (xy(1) - 1)^2 + (xy(2) - 2)^2;

% Define the initial guess for the optimization
% Initial guess for x and y
x0 = [0, 0];

% Use fminunc to find the minimum of the objective function

[optimalXY, optimalValue] = fminunc(objectiveFunc, x0)
```

```
%% Practice 2.1.


% 1. Define an utility function U(c) = 10c - c^2
% Evaluate it at x=10
% Use fminunc to find this maximum, starting from an initial guess  -10.

% 2. Define a welfare maximization problem for a social planner who aims
% to maximize the sum of utilities of 2 guys:

%       U1(x1) = -(x1 - c1)^2 and U2(x2) = -(x2 - c2)^2


% Assume c1 = 3 and c2 = 5.
% Use fminunc to find the optimal points x1 and x2, starting from an
% initial guess of x1 = 0 and x2 = 0.
```

```
%% 2.2. Anonymous Functions: Constrained Optimization


%Imagine we want to maximize f(x) = x^2 - 4x
% s.t.  x<=1.5

% We define the objective function as before
objectiveFunc = @(x) x^2 - 4*x; % Example quadratic function

% First Matlab needs to differentite between Inequality and Euqality Const
% And they have to be in Matrix Form:

% Inequality Constraints  A*x <= b  x:vector variables ;  b:vector scalars

A = 1;
b = 1.5;

% If the ineqaulity constraints are >= (x>=1.5), we negate it to have the
% equivalent:
% A= -1
% b = - 1.5;
% -x <= -1.5 which is equivalent to x >= 1.5

% Define the equality constraints Aeq*x = beq (if any [])
Aeq = [];
beq = [];

% Define the bounds of the variable lb <= x <= ub (if any [])
lb = [];
ub = [];


% Define the initial guess
x0 = 0;

% Use fmincon to find the minimum of the obj. function under constraints
[optimum, optimalValue]= fmincon(objectiveFunc, x0, A, b, Aeq, beq, lb, ub)


% In this example, the inequality is equivalent to a lower bound
% We define the objective function as before
objectiveFunc = @(x) x^2 - 4*x; % Example quadratic function
% Inequality Constraints
A = [];
b = [];
Aeq = [];
beq = [];
lb = [];
ub = 1.5;

x0 = 0;
[optimum, optimalValue]= fmincon(objectiveFunc, x0, A, b, Aeq, beq, lb, ub)
```

```matlab
% Complex Constrained Optimization Example

% f(x, y, z) = x^2 + y^2 - z    s.t.
% x + y <= 5
% y + z <= 6
% x + z = 3
% 0 <= x <= 4,
% 1 <= y <= 5,


% We'll negate the function as fmincon minimizes the function, and we want
% to maximize it
objectiveFunc = @(xyz) -(xyz(1)^2 + xyz(2)^2 - xyz(3));

% Initial guess for x, y, and z
x0 = [0, 0, 0];

% Inequality constraints (A*xyz <= b)
% Let's say we have the constraints
A = [1, 1, 0; 0, 1, 1];
b = [5; 6];

% Equality constraints (Aeq*xyz = beq)
% Suppose we also have the constraint
Aeq = [1, 0, 1];
beq = 3;

% Bounds on the variables (lb <= xyz <= ub)
% Let's set bounds on the variables:
lb = [0, 1, -Inf];
ub = [4, 5, Inf];

% Use fmincon to find the maximum of the obj. function under constraints
[optimalXYZ, optimalValue]=fmincon(objectiveFunc, x0, A, b, Aeq, beq,lb,ub)
optimalValue = -optimalValue % Negate back to get the maximum value


%% Practice 2.2.

% Assume an individual seeks to maximize their utility, which depends on
% two goods x and y, and labor l. The utility function is

%                U(x, y, l) = log(x) + log(y) + log(1-l)

% The individual's budget constraint is:

%                               p1*x + p2*y <= w*l

% where p1 and p2 are the prices of goods x and y, and w is the wage rate.
% The individual earns income by supplying labor, giving up leiure l.

% Find the combination of x, y, and l that maximizes utility subject to
% the budget constraint.

% Assume p1 = 2, p2 = 3, and w = 10. Also, the individual can work between
% 0 and 1.
```

```
%% 2.3. Anonymous Functions: Global Optimization

% The algorithms in matlab to find a global maximum, are based on
% finding a local maximum from a grid of initial points.
%% 3. Local Functions

% Local Functions can be used in two places:
% 1- Put it END OF FILE

multiplicationtable(1)


% 2- In another Script (name must match, same WD)

mu=5;
sigma=1;
x=normrnd(mu,sigma,1,100)

mainstats(x)

% NOTE You can make it print the results, but if you want to store
% the values, you need to declare ouput

[n, m, av, std] = mainstatsout(x)


% It might be required to introduce mulitple inputs
% n: elements of the set (number of letters)
% k: how many combinations of 3 letters we can have
calculateCombinations(4,2)

%% Practice 3.

% 1. Write at the end of script a function called 'displayFibonacci'
%    It take an integer 'n' as input and print the first 'n' numbers
%    After writing the script, call displayFibonacci(10);


% 2. Write a function in a separate script 'circleProperties.m'.
%    This function should take the radius of a circle as input and return
%    the area and the perimeter of the circle.
```

```
%% END. Local Functions

function multiplicationtable(number)

    disp(['Multiplication Table for ', num2str(number), ':']);
    for i = 1:10
        result = number * i;
        disp([num2str(number), ' x ', num2str(i), ' = ', num2str(result)]);
    end
end
```