```
%% A2 Process - Part 1
% Author: Pushpa
% Date: 9/04/2025
% Course: controlsystem
%
% Description:
% This script is a simple example that shows how to work with a "first-order process"
% in MATLAB. Think of it like a simple recipe. We'll set our "ingredients" (parameters),
% create our "recipe" (transfer function), and then "bake" it (plot the step response)
% to see what happens.

clc; % This command clears the command window below, so it's clean for our output.
clear; % This command clears all the variables in the workspace, so we start fresh.
close all; % This command closes any graph windows that are currently open.

%% Step 1: The ingredients (Given process parameters)
% We have three main ingredients for our process.
% They were found from a test, and they define how the process behaves.
K = 2.6; % K is the "gain." This tells us the final height (amplitude) of our graph.
T = 92; % T is the "time constant." This tells us how fast our graph rises.
L = 2; % L is the "time delay." This tells us how long our graph waits before it starts to rise.

%% Step 2: The recipe (Define the transfer function)
% We need to turn our ingredients into a single math formula that MATLAB can understand.
% This formula is called the "transfer function."
% The formula for our process is G(s) = K / (Ts + 1) * exp(-Ls).
s = tf('s'); % We first create 's', which is a special variable for transfer functions.
G = K / (Ts + 1) * exp(-Ls); % This line creates our main transfer function, G.

%% Step 3: Bake the first cake (Step response of the original process)
% Now we'll tell MATLAB to run our recipe and show us the result.
figure; % This command opens a new, empty window where our graph will be drawn.
step(G); % This command takes our recipe (G) and shows us its graph.
grid on; % This command adds a grid to the graph, which makes it easier to read numbers.
title('Step Response - Original Process'); % This adds a title to our graph.
xlabel('Time (seconds)'); % This labels the horizontal axis.
ylabel('Output'); % This labels the vertical axis.
legend('Original G(s)'); % This adds a small box that tells us what the line on the graph represents.

%% Step 4: Make a bigger cake (Multiply transfer function by 2)
% What if we want to double the final height of our graph?
% We can just multiply our entire recipe (transfer function) by 2.
```

```
G2 = 2 * G; % This creates a new transfer function, G2.
% It will have the same time constant and time delay, but the gain will be doubled.

%% Step 5: Compare the two cakes (Plot both step responses together)
% Let's see how our original graph compares to the new, bigger one.
figure; % This opens a brand new window just for this comparison graph.
step(G, G2); % This command plots both the original and the new transfer functions on the same graph.
grid on; % Again, we add a grid to make it easy to see.
title('Comparison of Original and Doubled Process');
xlabel('Time (seconds)');
ylabel('Output');
legend('Original G(s)', 'Doubled 2*G(s)'); % We label both lines so we can easily tell them apart.

%% Step 6: What's the final height? (Final amplitude)
% This last part shows the numbers for the final height of both graphs.
disp('—- Final Amplitudes —-'); % This prints a line of text in the command window.
% The final value is the same as the gain (K).
disp(['Final value of original system = ', num2str(K)]);
% The new final value is twice the original gain (2K).
disp(['Final value of doubled system = ', num2str(2K)]);
```