All scripts should start with author's name, date, and a statement of purpose

```
% Claude Lieber MD, 1/12/2022, Live Script to demonstrate Euler's method of
% solving differential equations
clear
close all
clc
```

Next all scripts should define constants (can be done in a separate script which is called by the main script) and variables. In Matlab, you do not need to define variables up front but in Python, C, C+, C++,.. you do need to define your variables initially. If a variable or constant is to be used in a separate function, it needs to declared as global

```
% Define Carrying Capacity
K=5e8;
% Define Birth rate
r=0.56;
% Define boundary condition: initial population
Pzero=120*3000;% cm^2 times #cells/cm^2
```

Decide on appropriate timestep, dt. In a later part of this exercise, you are asked to find the time when there are exactly (to 0.025%) a certain number of cells. The dt should be small enough that error is acceptable and in this case a dt that correlates to minutes or seconds would be convenient. There are 86400 sec/day. I chose a dt=1/86400

```
% Define dt
dt=1/86400;
% Define length of time the program should run, say 35 days
Tlast=35; % The nature of this problem, cell culture, means we are dealing with
days

% We are going to need a for/loop. The number of timesteps, iterations of the
loop,
% will be the total time divided by the timestep, dt
iterations=Tlast/dt;

% We are going to need a vector (array) to hold the value of P, the
% population, at each timestep. Call it Pall
Pall=zeros(1,iterations); % 1 row, iteration number of columns
```

```
% We will also want to track how fast the population grows, timestep by
% timestep. We create a vector dPall to hold this information
dPall=zeros(1,iterations);

% Creating this vector in advance speeds up the calculations. Otherwise,
% Matlab reconfigures the vector length with each timestep.This slows
computation

% Finally we will need to define the first value of the population, Pzero
P=Pzero;
```

Now we can create the for/loop to use the Euler method to solve the logistic population equation dP/dt=rP(1-P/K)

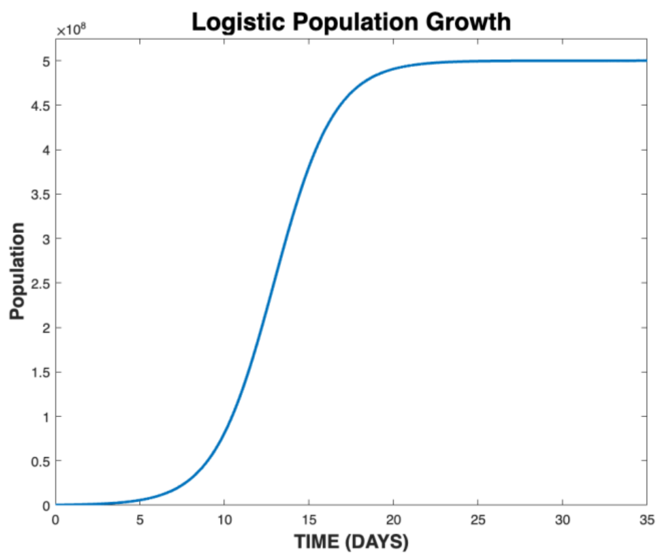
```
for k=1:iterations
    Pall(k)=P; % on the first iteration, Pall(1) will be Pzero,
    % A new value for Pall(k) will be entered for each iteration
    dP=r*P*(1-P/K); % This is our differential eq. -> the speed the population
grows
                    % with each timestep
    dPall(k)=dP; % saves this value of dp/dt (starting with P=P(0)) in our
vector
    P=dt*dP+P; %The new population value will be the last value of P
    % plus the rate it grows/timestep times the timestep. This
    % new value will be used at the start of each iteration
end
```

We have now saved all the the values for dP and P from time zero to 50 days. We need to plot these results. To do this, we need a time vector exactly the same length as the vectors Pall and dPall. The for/loop went from 1 to iterations. We want to plot from time zero, 0, to iteration-1

```
t=dt*(0:iterations-1); % creates time vector
```

Now to plot:

```
figure('color','w'); %creates a new figure with a white background
plot(t,Pall,'LineWidth',2) % plots population onto figure with a "thicker" line
xlabel('TIME (DAYS)', 'FontSize',14,'FontWeight','Bold')
ylabel('Population','FontSize',14,'FontWeight','Bold')
axis([0 35 0 K+0.25e8])
title('Logistic Population Growth','FontSize',18,'FontWeight','Bold')
```




```
figure('color','w');% creates new figure
plot(t,dPall,'linewidth',2) % plots progression of dp/dt over same timespan
xlabel('TIME (DAYS)', 'FontSize',14,'FontWeight','Bold')
ylabel('Rate of Change of Population','FontSize',14,'FontWeight','Bold')
axis([0 35 0 max(dPall)+0.25e7])
title('dP/dt: Logistic Population Growth','FontSize',18,'FontWeight','Bold')
```