Simulating population growth

Goal: define a function that generates a figure with curves for different rates of population growth over multiple generations, using the logistic growth model for population growth:

\[ p_{t+1} = r \cdot p_t \cdot \frac{(K - p_t)}{K} \]

- \( p_t \): current population
- \( p_{t+1} \): population in the next generation
- \( r \): growth rate
- \( K \): carrying capacity
**Guidelines & tips**

Define a function named `popGrowth` with four inputs:
- vector of growth rates to simulate (default [1.2 1.4 1.6 1.8 2.0])
- initial population (default 2)
- number of generations (default 25)
- carrying capacity (default 1000)

For each growth rate:
- create a vector to store the populations for each generation and store the initial population in the first location of the vector
- for each new generation, apply the formula to calculate the new population size and store it in the vector
- plot the populations for this growth rate

**New:** add input `maxPop`, replace inner `for` loop with a `while` loop that determines the number of generations needed to exceed `maxPop`

---

**Program complexity**

Designing large scale programs is fraught with peril

Divide, conquer and glue is a simple but powerful design strategy that helps us avoid danger
Tools of the trade

We have used functions and scripts to help divide problems into manageable chunks:

- lineFit, poleVault
- rotate, spin
- displayGrid, virus

What kinds of subtasks are performed by these individual functions in these programs, and ... ... why did we divide the programming task in this way?

Functions may...

- Perform a general function that’s useful in many contexts, e.g.
  - use lineFit function for any linear regression
  - use rotate function to rotate any figure

- Apply or test other functions, e.g.
  - poleVault tests the lineFit function

- Hide details of tasks like plotting or displaying data, e.g.
  - displayGrid displays current state of the virus
Functions help to avoid repetitious code

Consider a function with the following structure

```matlab
function outputs = myFunction (inputs)
    statements a
    statements b
    statements c
    statements b
    statements d
    statements b
end
```

Encapsulate repetitious statements in a separate function

Test, test, test!

“If there is no way to check the output of your program, in using that program, you have left the realm of scientific computation and entered that of mysticism, numerology, and the occult.”

Daniel Kaplan

*Introduction to Scientific Computation and Programming*
General tips on testing

Test and debug each function on its own

Create test data for simple cases where expected intermediate results and final answer can be easily verified

Be thorough! Construct examples to test all cases considered by your program

Functions versus scripts

Functions usually have one or more inputs that provide data or control aspects, and one or more outputs

Scripts perform a specific set of actions and do not have inputs or outputs

Execution of a function creates a private, temporary environment of variables 😊

Scripts have access to variables defined in the environment within which the script is called*

* Danger Will Robinson!!!
Subfunctions

An M-file can only contain one function that can be called from the Command Window or from another code file. This function must be placed at the beginning of the file and its name must be the same as the file name. Other subfunctions can be defined in an M-File, but can only be called by functions in the same M-File.

Subfunctions for a ferris wheel movie

```matlab
function ferrisWheel
% displays an animation of a rotating ferris wheel
for frame = 1:36
    drawBase;
    hold on
    spokeCoords = drawWheel(10*frame);
    drawCars(spokeCoords);
    hold off
end

function drawBase
% draw the blue base of the ferris wheel

function spokeCoords = drawWheel (angle)
% draw the black spokes at the input angle and return
% the coordinates of the endpoints of the spokes

function drawCars (spokeCoords)
% draw a colored car at each location in spokeCoords
```