Assessment 5: Population Carrying Capacity

CSCI 128

Problem

In this assessment, we will be investigating a dynamical system that models the growth and contraction of a population as a result of competition.

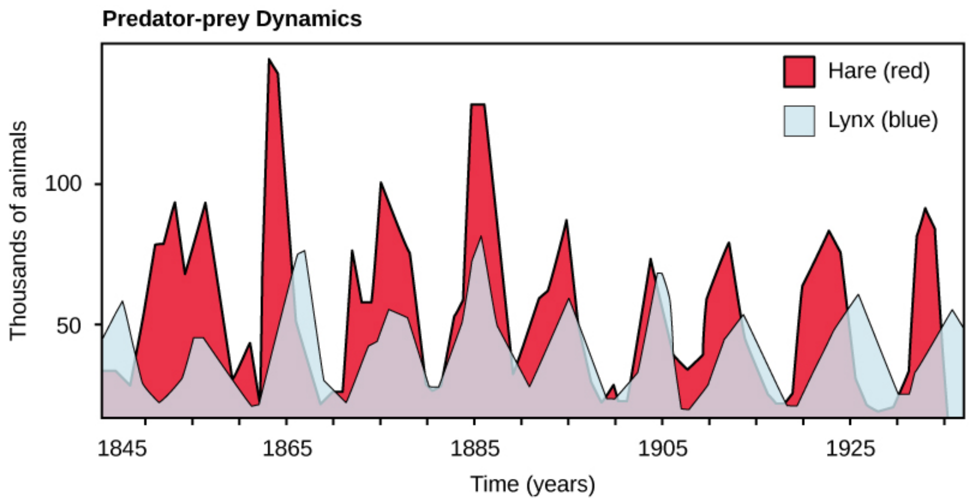

Figure 1: Hare and Lynx population dynamics over time. Figure source.

Consider a one-dimensional discrete time model for the carrying capacity of a population that is defined by the following equation:

$$x_{n+1} = \lambda * x_n(1 - x_n)$$

x_n represents the proportion of the population as a fraction ($0 \leq x_n \leq 1$) of the carrying capacity at time n. λ is the growth term ($0 < \lambda \leq 4$) that governs the growth of the population at each time step. $1 - x_n$ is the competition term that limits the population growth as x_n increases.

For certain values of λ this system **may**, after (several) iterations, converge to a single value $x_{n-1} = x_n$ for all x_n after convergence.

However, sometimes this system converges into a cycle, (e.g. 0.5,0.8,0.5,0.8,...).

Input

The first line of input will contain a decimal number, the starting population x_0 and the second line will contain λ, again a decimal number. It is guaranteed that λ will follow the limits described in the previous section, $0 < x_0 < 1$ and the inputs will be given accordingly that the result will converge into a single number.

Output

If the value converges only to a single value, the first and only output is the value/limit to which it converges to **six** decimal places.

If the value converges only to a cycle, print all the values in the cycle as the output to **six** decimal places.

For the sake of this assessment, "converges" means two x_n values are equal when rounded up to **six** decimal places. This applies for both converging to single values **and also converging to cycles.**

Strategy

1. For the purposes of this assessment, round the floating point values to 6 decimal places before comparing them.
2. There is a certain continuous range of λ for which the population diverges to chaos. You are not responsible of handling this case and autograder the test cases will not cover this range. However, be aware that if your program takes too long to run or crashes, you may have given such a value of lambda. In that case test your program with different combinations of values. If your program responds correctly in the above mentioned test cases, you can feel confident about your code.

The following pseudocode may help with the implementation of your program:

Algorithm 1 Population Carrying Capacity

Input: x_0 (initial fraction) , λ (growth term) **Output:** convergence point(s)

```
$x = x_0$
while Not Converged do
    $x_{new} = \lambda * x * (1 - x)$
    $x_{new}$ =round $x_{new}$ to 6 decimal places
    if $x_{new} = x$ then
        print($x_{new}$)
        exit the loop
    else if $x_{new}$ is previously visited then
        print the values in the cycle up to $x_{new}$
        exit the loop
    end if
    $x = x_{new}$
end while
```

Reflection

In class this week we discussed debugging strategies. These will only become more useful as you begin to write more complicated programs and as you encounter semantic errors more frequently over simpler syntax and runtime errors.

What debugging strategies did you employ to complete this assignment? These can be the same methods we discussed in class or others. What problems in your original code were you able to resolve thanks to those debugging strategies? It retrospect, were there other debugging strategies you could have used that might have been more effective?

Unlikely as it is, if you did not need to do any debugging, then what about your development process do you think was so effective that you were able to write correct code in the first attempt?

Sample Executions

Example 1

```
INPUT> 0.2
INPUT> 2.343

OUTPUT 0.573197
```

Example 2

```
INPUT> 0.7
INPUT 3.2

OUTPUT 0.799455
OUTPUT 0.513045
```

Example 3

```

INPUT> 0.9
INPUT> 3.1

OUTPUT 0.764565
OUTPUT 0.558017
```

Example 4

```

INPUT> 0.1
INPUT> 2.111

OUTPUT 0.526291
```