Computer Science Courses

PROJECT 2: FOXES AND RABBITS

Project due date: October 1st 23:59PM

This is an individual project.

The project is worth 100 points.

Introduction

One of your friends studying demography asks you to help him write a program to study the dynamics of biological systems in which two species interact, one being a predator and one its prey, as, for example, foxes and rabbits.

He expects that not only can the simulator calculate the evolution of the the two species population based on his models, but also can visualize the results.

Background information & material

Background information

Look at An example problem to get more information about the Lotka-Volterra model and to get an idea of what the visualization of the population growth can tell you.

Material provided to you

To facilitate your work we provide to you some skeleton code in the file project2-sk.py:

- the **drawCurve()** function code
- comments aimed at helping you in writing the code for the TODOs defined later.

SETUP

Under your CS177 folder create the folder project2.

Download there the skeleton code (project2-sk.py) found Here

PROBLEM DESCRIPTION

One of your friends studying demography asked you to help him write a population growth simulation program, such that he can use it to test his model describing how the population of two competing species, foxes and rabbits, changes in a given region. He expected that not only can the simulator calculate the population growth based on his models, but also can visualize the results.

The overall structure of the simulation program that you are asked to write will be composed of the following parts:

1. a part that will get from the user the needed input parameters (such as the values of the parameters of the model, the number of steps that each simulation run perform). This part will be implemented in the main() function (see TODO 3).

2. the part that performs the calculations of the equations of the model (see TODO1 and TODO 2)
3. the part the perform the graphical visualization (n 2D) of the values calculated by the equations above.

Visualization is a powerful tool to get a deeper and immediate understanding of the phenomenon under investigation. The visualization of the plot of the populations' allows you to get a deeper and immediate understanding of how the rabbits and foxes populations' growth are "intertwined'. Visualization can be achieved in Python by using the functions provided by the following libraries:

- numpy: this library provides functions to perform numerical calculations
- pylab: this a library that performs numerical calculation and visualization

We provided to you in the skeleton code the function **drawCurve()** that will draw the plot of the populations' growth. You need to call it at the right place in the **main()** function passing the right parameters.

NOTE

- The numpy and pylab libraries can be found at: http://www.lfd.uci.edu/~gohlke/pythonlibs/ [http://www.lfd.uci.edu/~gohlke/pythonlibs/]
- These libraries are not currently available for Python3.2 for Macintosh or Unix systems. If you fall into this realm you will need to develop your program in the laboratory.

The model

To describe the evolution of the two interacting species populations, your friend wants to use the Lotka–Volterra equations, also known as the predator–prey equations.

In this model:

- x represent the number of prey (for example, rabbits);
- y represent the number of some predator (for example, foxes);
- dy/dt and dx/dt represent the growth rates of the two populations over time;
- t represents the time

The growth of the two populations is described by the equations:

```
dx/dt = x(a-by)
dy/dt = -y(g-dx)
```

where a,b,d, and g are parameters of the model. Namely:

- a is the growth rate of prey, i.e. of rabbitts,
- b is the rate at which predators, i.e. foxes, destroy prey,
- g is the death rate of predators (also predators die…),
- d is the rate at which predators increase by consuming the prey

NOTES

The two mathematical equations above describe the algorithm that governs the rabbit and foxes population growth in the Lotka-Volterra model.

TODO # 1 (15 points)

Write the function deltaX that calculates the increment of the rabbit population according to the model.

1. This function should take 4 input arguments:
    - x the current rabbit population
    - y the current fox population
    - a is the growth rate of the rabbit population without predation.
    - b is the rate at which the foxes eat the rabbits.

NOTE: this function will generate the change in the rabbit population. You will need to reflect this change in the variable for the rabbit population. Having computed the current change, you would update the number of rabbits by adding the change, so obtaining the population at the next time step. This is an Euler schema for solving the equation.

TODO # 2 (15 points)

Write the function deltaY that calculates the increment of the foxes population according to the model.

1. This function should take 4 input arguments:
    - x the current rabbit population
    - y the current fox population
    - g the death rate of foxes.
    - d the rate at which fox population increases by consuming the rabbits

NOTE: this function will generate the change in the fox population. You will need to reflect this change in the variable for the fox population. Update the number of foxes in the same manner as you did for the rabbits in TODO #1.

TODO #3**(40 Points)

Write the simulation function. This function will ask the user for the number of runs and then execute that number of iterations of the model. The simulation should run as follows:

1. Ask user for number of runs, and save it in a variable
2. Create a list for rabbits and a list for foxes
3. Repeat for each run:
    a. calculate the increment of rabbits population by calling **deltaX()** function and save this in a variable
    b. calculate the increment of foxes population by calling **deltaY()** function and save this in a variable
    c. update current rabbit and fox population
    d. add the newly calculated current population of rabbits and foxes to the two lists
4. After all iterations of the simulation run are completed, print out the rabbits and foxes minimum population
5. Call the drawCurve() function

Please look at the grading rubric for the score assigned to each of the five items above.

Other Requirements

- You must calculate the two population increments one immediately after the other
- The population is considered to be extinct if it falls below 0.001. In this case you must set the current population to zero. Be sure to add in checks during each iteration of the loop.
- You must also keep track of the minimum of each population. You will have to do this in the loop.
- The drawCurve() function has three arguments as input. The first two are the the foxes and rabbits list (in that order). The third argument is the number of runs as input by the user. If you are unfamiliar with lists, see the list explanation below.

Lists

Imagine a grocery list written on a piece of paper. A list in Python is organized the same way. A sequential ordering of data. can

add items to the list using myList.append(). Sample code is provided below. You can also reference the python documentation here: http://docs.python.org/tutorial/datastructures.html [http://docs.python.org/tutorial/datastructures.html]

```
myList = []  # list definition

# Adding new items to the list using "append"
myList.append(50)  # add fist item
myList.append(100) # add second item

print(myList)
>>>[50,100]        # two items in the list
```

TODO #4** (30 points)

To make the population growth simulator easier to use by your friend, who knows nothing about programming, you need to create a user interface. You can create it in the **main()** function. The user interface should include the following menu:

1. A way to set each parameter of the model(a,b,g,d)
2. A way to set the initial number of rabbits and foxes
3. A way to set all parameters at once
4. A way to start the simulation
    a. the user will define (by input) the number of rounds the simulation will run
    b. after the simulation is complete the program will pause and display a graph output. After the graph is closed, the simulation will return to the main menu.
5. A way to exit the program
    a. the program will not end until the exit option is chosen.

The user interface should be done as a text menu, which looks similar to:

```
              Main Menu
_____________________________________________
1)Set the growth rate of rabbits.
2)Set the rate at which foxes kill rabbits.
3)Set the death rate of foxes.
4)Set the rate at which foxes increase.
5)Set initial number of rabbits and foxes
6)Set all parameters at once.
7)Run Simulation
8)Exit
Please choose option:
```

When the user inputs a number the appropriate action will be taken. After the action has completed the program should return to the menu and request additional input.

Note

1. This menu will require a while loop that will keep running until the user chooses to exit.

TEST CASES

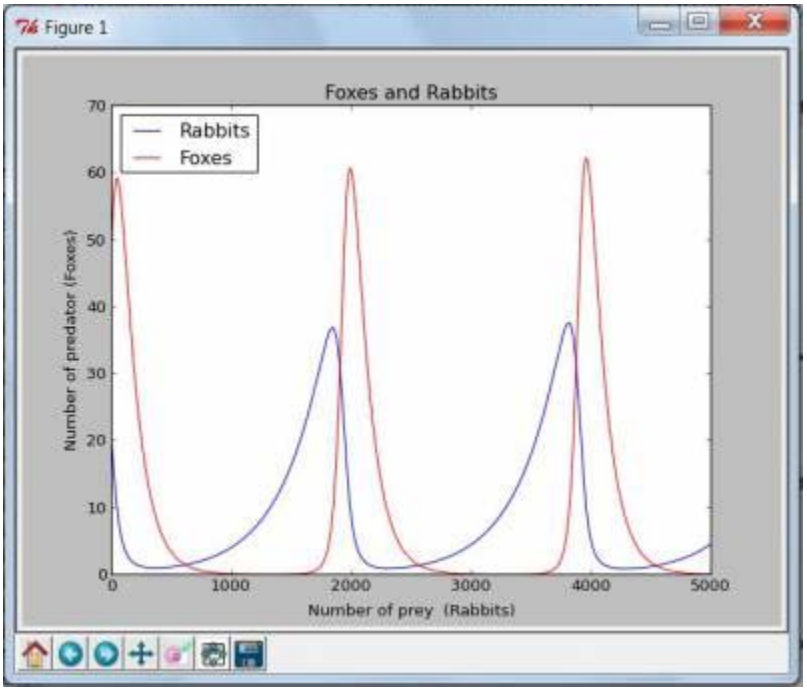
To verify that your program is performing as expected, use the test cases detailed in the followin.

Test Case 1:

```
a,b,g,d = 0.0028, 0.00032, 0.008, 0.0008
starting number of rabbits: 20
starting number of foxes: 50
number of runs: 5000
```

```
>>>
min fox population: 0.055052 , min rabbit pop: 0.926731
```

Here the populations should exhibit cyclical patterns with the number of foxes peaking near 2000 and 4000 steps. This is the normal boom-bust cycle.

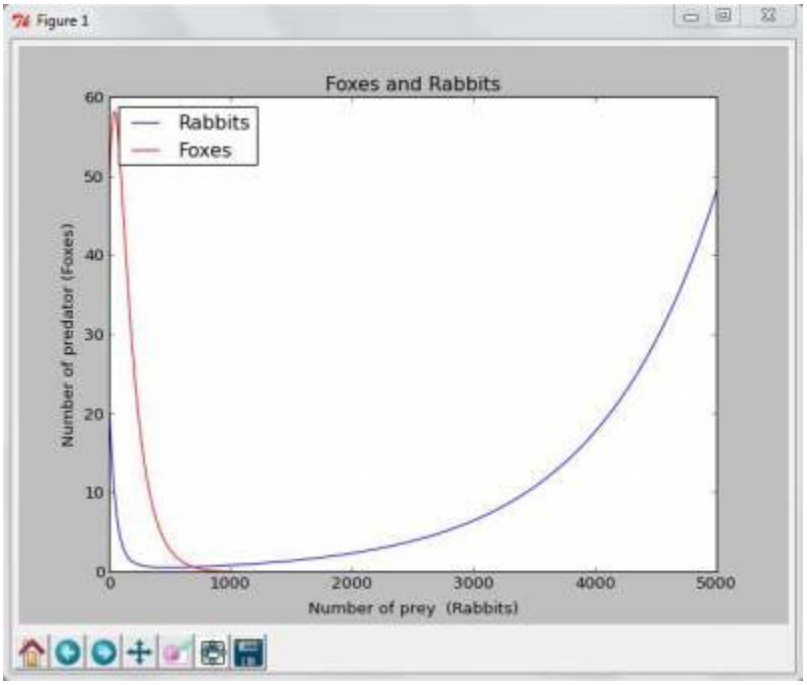


Test Case 2:

```
a,b,g,d = 0.001, 0.00032, 0.008, 0.0008
starting number of rabbits: 20
starting number of foxes: 50
number of runs:5000

>>>
min fox population: 0.000000 , min rabbit pop: 0.609701
```

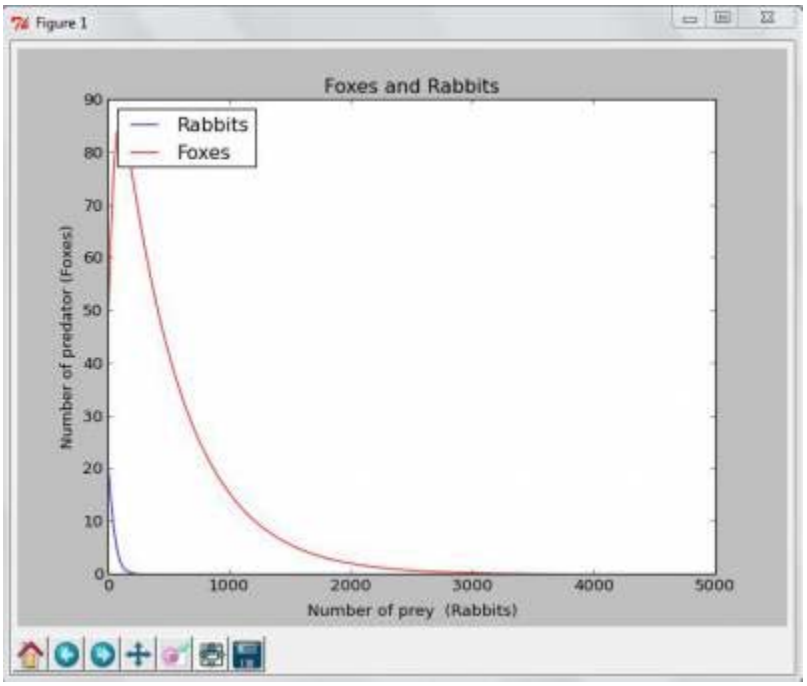
Here the rabbits are harder to catch and the foxes die out from hunger near step 1600.



Test Case 3:

```
a,b,g,d = 0.0028, 0.00032, 0.002, 0.0008
starting number of rabbits: 20
starting number of foxes: 50
number of runs:5000
>>>
min fox population: 0.005104 , min rabbit pop: 0.000000
```

Now the foxes live longer, thus predation is more severe. The rabbits die out around step 580. Since they have lost their food source, foxes will die out too at a later time…



Grading Rubric

TODO1 15 Points

TODO2 15 Points

TODO3 40 Points

1. 3 Points
2. 7 Points
3. 20 Points
4. 5 Points
5. 5 Points

TODO4 30 Points

1. 6 Points
2. 6 Points
3. 6 Points
4. 6 Points
5. 6 Points

TURNIN INSTRUCTIONS

Login into your UNIX CS account.

```
$cd CS177
$cd project2
```

Then launch the following command AS IS:

```
$turnin -v -c cs177=COMMON -p project2 project2
```

Note This procedure has changed slightly from the last project. **DO NOT** use any section numbers when calling turn-in. Please run the command as shown above!

---

cs17700/projects/project2.txt · Last modified: 2012/09/28 03:25 by rglasser