

## PART B - PROBLEM 3-1: RUNNING AND ANALYZING A SIMPLE SIMULATION (NO DRUG TREATMENT)

(10/10 points)

You should start by understanding the population dynamics before introducing any drug.

Fill in the function `simulationWithoutDrug(numViruses, maxPop, maxBirthProb, clearProb, numTrials)` that instantiates a `Patient`, simulates changes to the virus population for 300 time steps (i.e., 300 calls to `update`), and plots the average size of the virus population as a function of time; that is, the x-axis should correspond to the number of elapsed time steps, and the y-axis should correspond to the average size of the virus population in the patient. The population at `time=0` is the population after the first call to `update`.

Run the simulation for `numTrials` trials, where `numTrials` in this case can be up to 100 trials. Use pylab to produce a plot (with a single curve) that displays the average result of running the simulation for many trials. Make sure you run enough trials so that the resulting plot does not change much in terms of shape and time steps taken for the average size of the virus population to become stable. **Don't forget to include axes labels, a legend for the curve, and a title on your plot.**

You should call `simulationWithoutDrug` with the following parameters:

- `numViruses` = 100
- `maxPop` (maximum sustainable virus population) = 1000
- `maxBirthProb` (maximum reproduction probability for a virus particle) = 0.1
- `clearProb` (maximum clearance probability for a virus particle) = 0.05

Thus, your simulation should be instantiating one `Patient` with a list of 100 `SimpleVirus` instances. Each `SimpleVirus` instance in the viruses list should be initialized with the proper values for `maxBirthProb` and `clearProb`.

[Hint: graphing](#)

Consult [reference documentation on pylab](#) as reference. Scroll down on the page to find a list of all the [plotting commands](#) in pylab.

[Hint: testing your simulation](#)

For further testing, we have provided the .pyc (compiled Python) files for the completed `Patient` and `SimpleVirus` classes (and for Problem 5, the `ResistantVirus` and `TreatedPatient` classes) that you can use to confirm that your code is generating the correct results during simulation.

If you comment out your versions of these classes in `ps3b.py`, and add the following import statements to the top of your file, you can run the simulation using our pre-compiled implementation of these classes to make sure you are obtaining the correct results. This is a good way to test if you've implemented these classes correctly. Make sure to comment out the import statement and uncomment your implementations before moving to Problem 4.

For Python 2.7:

```
from ps3b_precompiled_27 import *
```

```

1 # Enter your definition for simulationWithoutDrug in this box
2 def simulationWithoutDrug(numViruses, maxPop, maxBirthProb, clearProb,
3                             numTrials):
4     """
5     Run the simulation and plot the graph for problem 3 (no drugs are used,
6     viruses do not have any drug resistance).
7     For each of numTrials trial, instantiates a patient, runs a simulation
8     for 300 timesteps, and plots the average virus population size as a
9     function of time.
10
11     numViruses: number of SimpleVirus to create for patient (an integer)
12     maxPop: maximum virus population for patient (an integer)
13     maxBirthProb: Maximum reproduction probability (a float between 0-1)
14     clearProb: Maximum clearance probability (a float between 0-1)
15     numTrials: number of simulation runs to execute (an integer)
16     """

```

Correct

## Test results

Hide output

**CORRECT**

Test: simulation 1

Note: your simulation should only be using the plot, title, xlabel, ylabel, legend, figure, and show functions from the PyLab module.

Additionally, be sure you call `pylab.show()` last. The order of other lines do not matter.

Output:

[illegible]

Test: simulation 2

Additionally, be sure you call `pylab.show()` last. The order of other lines do not matter.

```
Test: simulationWithoutDrug(100, 200, 0.2, 0.8, 1)
Plotting...
Plotting values: [84.0, 47.0, 33.0, 17.0, 9.0, 10.0, 8.0, 3.0, 1.0, 1.0, 0.0, 0.0, 0.0, 0.0,
0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0,
0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0,
0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0,
0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0,
0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0,
0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0,
0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0,
0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0,
0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0,
0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0,
0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0,
0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0,
0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0]
Successfully called pylab.xlabel with label: Time
Successfully called pylab.ylabel with label: Population average size
Successfully called pylab.title with label: Simulation: Population size with time
Successfully called pylab.legend
Successfully called pylab.show
```

Additionally, be sure you call `pylab.show()` last. The order of other lines do not matter.

```


Test: simulationWithoutDrug(1, 90, 0.8, 0.1, 1)
Plotting...
Plotting values: [2.0, 4.0, 7.0, 9.0, 17.0, 27.0, 36.0, 47.0, 58.0, 64.0, 72.0, 83.0, 90.0,
88.0, 90.0, 85.0, 86.0, 84.0, 90.0, 88.0, 88.0, 81.0, 85.0, 85.0, 86.0, 88.0, 87.0, 85.0,
87.0, 90.0, 89.0, 86.0, 82.0, 88.0, 84.0, 84.0, 89.0, 87.0, 86.0, 84.0, 87.0, 89.0, 86.0,
85.0, 79.0, 87.0, 89.0, 87.0, 89.0, 85.0, 86.0, 81.0, 85.0, 85.0, 82.0, 83.0, 85.0, 85.0,
81.0, 91.0, 88.0, 85.0, 86.0, 83.0, 86.0, 90.0, 85.0, 86.0, 87.0, 85.0, 87.0, 87.0, 94.0,
89.0, 88.0, 84.0, 84.0, 76.0, 72.0, 83.0, 91.0, 89.0, 89.0, 86.0, 87.0, 92.0, 88.0, 85.0,
79.0, 82.0, 90.0, 90.0, 85.0, 86.0, 89.0, 83.0, 87.0, 84.0, 85.0, 85.0, 85.0, 88.0, 92.0,
90.0, 87.0, 91.0, 87.0, 84.0, 85.0, 84.0, 88.0, 90.0, 89.0, 89.0, 87.0, 85.0, 82.0, 82.0,
84.0, 94.0, 86.0, 89.0, 85.0, 81.0, 79.0, 80.0, 75.0, 77.0, 84.0, 85.0, 88.0, 85.0, 85.0,
84.0, 81.0, 85.0, 84.0, 80.0, 83.0, 87.0, 80.0, 78.0, 81.0, 79.0, 85.0, 87.0, 86.0, 85.0,
87.0, 81.0, 83.0, 83.0, 78.0, 83.0, 86.0, 86.0, 86.0, 88.0, 84.0, 85.0, 82.0, 86.0, 80.0,
83.0, 86.0, 89.0, 88.0, 86.0, 83.0, 80.0, 87.0, 96.0, 88.0, 88.0, 87.0, 85.0, 83.0, 88.0,
89.0, 83.0, 86.0, 84.0, 82.0, 82.0, 89.0, 86.0, 82.0, 82.0, 86.0, 91.0, 90.0, 88.0, 91.0,
88.0, 81.0, 84.0, 80.0, 80.0, 90.0, 84.0, 80.0, 86.0, 83.0, 84.0, 87.0, 88.0, 87.0, 85.0,
90.0, 85.0, 84.0, 86.0, 90.0, 83.0, 80.0, 82.0, 86.0, 88.0, 89.0, 86.0, 79.0, 86.0, 86.0,
87.0, 85.0, 88.0, 88.0, 81.0, 77.0, 81.0, 86.0, 96.0, 88.0, 90.0, 84.0, 85.0, 89.0, 88.0,
89.0, 85.0, 83.0, 86.0, 86.0, 87.0, 85.0, 91.0, 87.0, 83.0, 83.0, 95.0, 89.0, 92.0, 88.0,
86.0, 85.0, 84.0, 85.0, 85.0, 83.0, 84.0, 85.0, 87.0, 85.0, 83.0, 87.0, 86.0, 86.0, 86.0,
86.0, 91.0, 90.0, 85.0, 82.0, 79.0, 82.0, 86.0, 84.0, 85.0, 84.0, 91.0, 90.0, 83.0, 85.0,
76.0, 84.0, 83.0, 88.0, 88.0, 85.0, 87.0, 89.0, 88.0, 86.0, 89.0, 87.0, 89.0, 83.0, 82.0,
84.0, 80.0]
Successfully called pylab.xlabel with label: Time
Successfully called pylab.ylabel with label: Population average size
Successfully called pylab.title with label: Simulation: Population size with time
Successfully called pylab.legend
Successfully called pylab.show

```

[Check](#)[Save](#)*You have used 3 of 30 submissions*

## PART B - PROBLEM 3-2 (5/5 points)

A good question to consider as you look at your plot is: about how long does it take before the population stops growing?

- ☐ About 20 time-steps
- ☐ About 80 time-steps
- ☒ About 150 time-steps 
- ☐ About 200 time-steps
- ☐ About 250 time-steps

[Check](#)[Save](#)*You have used 1 of 10 submissions*[Show Discussion](#)[New Post](#)

EdX offers interactive online classes and MOOCs from the world's best universities. Online courses from MITx, HarvardX, BerkeleyX, UTx and many other universities. Topics include biology, business, chemistry, computer science, economics, finance, electronics, engineering, food and nutrition, history, humanities, law, literature, math, medicine, music, philosophy, physics, science, statistics and more. EdX is a non-profit online initiative created by founding partners Harvard and MIT.

© 2014 edX, some rights reserved.

[Terms of Service and Honor Code](#)

[Privacy Policy \(Revised 4/16/2014\)](#)

### About & Company Info

[About](#)[News](#)[Contact](#)[FAQ](#)[edX Blog](#)[Donate to edX](#)[Jobs at edX](#)

### Follow Us

[Twitter](#)[Facebook](#)[Meetup](#)[LinkedIn](#)[Google+](#)