COMP 200 & COMP 130
Assignment 2: Control Flow & Predator-Prey

Be sure to read the course policies, as posted on the course web site:

`http://www.clear.rice.edu/comp200/policies.shtml`.

Work individually on this assignment. Turn in two files: A Python file, *`netid`*`_2.py`, with all your code for the programming problems. A text file in Adobe Acrobat or Microsoft Word format, *`netid`*`_2.pdf/doc/docx`, with all your text for the non-programming problems. When submitting, double-check that your files are indeed uploaded.

Total points: 100 for COMP 200, 150 for COMP 130.

Wherever reasonable, use functions that have been defined previously on this assignment, on a previous assignment, or in class.

For all students (75 points total)

More statistics (20 points total)

1. (20 points) In the previous homework, you defined the functions `arithmeticMean` and `stdDeviation` to calculate the arithmetic mean and standard deviation, respectively, of a non-empty list of numbers. One use of these values is to then look at how many of the original values are within, say, 1 or 2 standard deviations of the mean, and to compare this against the statistically expected result.

   Define a function `withinNStdDeviations` that takes two arguments: a non-empty list of numbers, and another number n. It returns a count of the number of values in the list that are within n standard deviations of the data's arithmetic mean.

Get Your Ducks in a Row (55 points total)

2. (20 points) Given a list, we want to know if it is sorted in ascending order. I.e., is each element no smaller than the one that precedes it. Thus, `[2,4,7,10]` and `["cat","dog","dog","duck","snake"]` are each sorted, whereas `[3,7,5,10]` is not. Additionally, the empty list `[]` and any one-element list such as `[3]` or `["cat"]` is sorted.

   Define a function `isSorted` that takes a list and returns a Boolean indicating whether the list is sorted.

   You are *not* allowed to use the built-in `list.sort()` or `sorted()`.

Given a list, we would often like to sort it. While Python has this operation built-in, we now have the tools to do it for ourselves.

3. (20 points) Given a list that is sorted in ascending order and another value, we want to add the value into the appropriate location in the list so that it is still sorted.

   Define a function `insertSorted` that behaves as follows:

```
numbers = [2,4,8,9]

insertSorted(numbers,5)
print numbers     # Should print [2,4,5,8,9]

insertSorted(numbers,5)
print numbers     # Should print [2,4,5,5,8,9]

insertSorted(numbers,10)
print numbers     # Should print [2,4,5,5,8,9,10]

numbers = []

insertSorted(numbers,5)
print numbers     # Should print [5]
```

Hint: Either use `list.insert(index,value)` to add the value at that index (i.e., position) or use `list.append(value)` to add it at the end.

You are *not* allowed to use the built-in `list.sort()` or `sorted()`.

4. (15 points) Define a function `sort` that sorts a list using the idea illustrated by this video:

   `http://www.youtube.com/watch?v=ejpFmtYM8Cw`.

   We create a new list, repeatedly insert (in sorted order) each value from the input list into this new list, and finally return the result.

   You are *not* allowed to use the built-in `list.sort()` or `sorted()`.

As a useful tip, observe that we could use our `isSorted` function to do *some* of our testing on our `insertSorted` and `sort` functions. It won't check if we get the correct result, but it will check an important desired property of the result. For example,

```
numbers = [2,4,8,9]

insertSorted(numbers,5)
if not isSorted(numbers):
    print "Incorrect result: ", numbers

insertSorted(numbers,5)
if not isSorted(numbers):
    print "Incorrect result: ", numbers

insertSorted(numbers,10)
if not isSorted(numbers):
    print "Incorrect result: ", numbers

numbers = []

insertSorted(numbers,5)
if not isSorted(numbers):
    print "Incorrect result: ", numbers
```

For COMP 200 students (25 points total)

Predator/prey

5. (25 points total) Pick three different sets of input rates for the predator-prey model.

   As described in class, as you increase the number of time steps per year, the accuracy of the implementation improves. In the following steps, you will investigate this improvement further.

   (a) (5 points) For each input set, make time plots for 1, 10, 100, and 1000 steps per year.

   (b) (10 points) As you increase the number of steps, the plots converge to the ideal result. Experiment to find roughly how many steps (any value – not just those from the previous part) until no visible change in plots results. For each input set, report this number of steps and display the plot.

   (c) (10 points) What pattern(s) do you see across all of the input sets?

For COMP 130 students (75 points total)

Note: Your code should be set up such that simply running your submitted file should automatically generate all the requested plots on the screen without any manual intervention to change variable values, comment or uncomment code, etc. It would be best if all the plots showed at once so that they can easily be compared (just call `show()` once, at the end of the file).

Also, you are welcome to use the plotting functions from the `plotfuncs.py` file from the COMP 200 Resources site, and you are free to modify that code to suit your needs. You **must** submit your `plotfuncs.py` file, modified or not, along the rest of your assignment!

Second-Order Lotka-Volterra Implementation (35 points total)

Suggestion: If you add a multiplicative factor to the second-order term, called, say `incl2ndOrder`, that is either set to the value of 1 or 0, you can easily turn the second-order correction on and off.

6. (a) (15 points) Implement the Lotka-Volterra algorithm using a second-order Taylor expansion approximation.

   (b) (5 points) Make time plots showing both first- and second-order algorithm results to demonstrate the increased accuracy of your second-order result. Your plots should show the effects of changing the time steps in the first-order approximation.

   (c) (5 points) Make predator population vs. prey population "orbit" graphs showing the increased accuracy of the second-order vs. first-order results. As in the time plots, the effects of changing the time steps in the first order approximation should be shown. It would be best to use the same data calculated for the time plots so that you can easily compare them.

   (d) (10 points) In your text file, explain the differences between the first- and second-order graphs in parts 6b and 6c and why you think those differences arose. Support your assertions with references to concrete and specific features in your plots. Simple answers such as "The second order approximation is more accurate." will not be accepted because this short answer does not explain how that increased accuracy manifests itself in the shape of the graphs.

Prey Starvation Effects (40 points total)

Suggestions: Use a function to calculate the "max" value since the "cutoff" value is being changed often. Make your function such that the ratio of max to cutoff can be controlled.

7. (a) (10 points) Implement **both** the linear and quadratic starvation models by using an `if-else` conditional in your second-order Taylor expansion Lotka-Volterra algorithm that selects one model vs the other based on a supplied Boolean value. You may use the same multiplicative factor technique as the previous problem to include or not include the entire starvation effect in the calculation.

   (b) (5 points) Make time and orbit plots to demonstrate the differences between the models at the following cutoff values. As in class, set the max value to be twice the cutoff value.

   - Cutoff population is 75% of the initial prey population.
   - Cutoff population is 100% of the initial prey population.
   - Cutoff population is 150% of the initial prey population.
   - Cutoff population is 500% of the initial prey population.

   (c) (10 points) Based on the shape of starvation effect rates as a function of the prey population, explain the differences you see in the results for the linear vs. quadratic models. Be sure to support your assertions with references to concrete and specific features in your plots.

   (d) (5 pts) Create time and orbit plots of your linear model, where you change the max value to

   - 1.5 times the cutoff
   - 2.0 times the cutoff
   - 5.0 times the cutoff
   - 20.0 times the cutoff

   (e) (10 pts) Explain your part 7d results by relating them to your answer in part 7c. How does changing the max value in the linear starvation model relate the shape of the quadratic model? (Hint: Draw a picture of the linear model for various max values.)

For all students

Feedback

1. Roughly how many hours did you spend on the two sets of finger exercises?
2. On a scale of 1 (very easy) to 5 (very difficult), how difficult were the finger exercises?
3. Roughly how many hours did you spend on this homework?
4. On a scale of 1 (very easy) to 5 (very difficult), how difficult was this homework?
5. Which material did you find most challenging?
6. Did you feel that the class material adequately prepared you for the homework?