COMP 200 & COMP 130
Assignment 1: Computational Thinking & Introductory Python

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, *`netid1`*`_1.py`, with all your code for the programming problems. A text file in Adobe Acrobat or Microsoft Word format, *`netid1`*`_1.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.

For all students

We're going to the movies! (45 points total)

Our friend owns a small one-screen movie theater and recently ran an experiement to determine the relationship between the price of a ticket and attendance. At a price of $10.00 per ticket, 200 people attend a performance. For each decrease in price by a quarter ($.25), the attendance increases by 15. Unfortunately, the increased attendance also comes at an increased cost in royalties and overhead. Every performance costs the owner $1000, plus five cents ($0.05) per attendee.

The owner would like to know the exact relationship between profit and ticket price so that he can determine the price at which he can make the highest profit.

While the task is clear, how to go about it is not. All we can say at this point is that several quantities depend on each other. When we are confronted with such a situation, it is best to tease out the various dependencies one at a time:

- *Profit* is the difference between revenue and cost.
- The *revenue* is exclusively generated by the sale of tickets. It is the product of the ticket price and the number of attendees.
- The *costs* consist of two parts: a fixed part ($1000) and a variable part that depends on the number of attendees.
- Finally, the problem statement also specifies how the number of attendees depends on the ticket price.

2. (15 points) How many attendees can afford a show at a ticket price of $6.00, $8.00, and $10.00? What are the total costs, revenue, and profit for each show at these same prices?
3. (15 points) Define a mathematical equation for each of the following: the number of attendees, the costs, revenue, and profit. Your equations should refer each other where possible, without being circular.
4. (15 points) Define the corresponding Python functions: `attendees`, `costs`, `revenue`, and `profit`. Each function has one argument, the ticket price. Again, your functions should call each other where possible, without being circular.

Statistics (25 points total)

In science, engineering, and social science, it is very common to generate sample data and analyze it statistically. Python does not have statistical tools built-in, but we can define many of them with simple functions on lists.

4. (5 points) The *arithmetic mean*, or average, of a collection of numbers is the sum of the numbers divided by the count of numbers. For example, the mean of $[3, 7, 1, 2, 10]$ is $(3+7+1+2+10)/5 = 4.6$.

   Define a function `arithmeticMean` that takes a non-empty list of numbers and returns its arithmetic mean. Be sure to use the relevant functions from the finger exercises.

5. (15 points) The *variance* of a collection of numbers is the mean of the squared deviation (difference) of each number from the collection's mean. For example, the variance of $[3, 7, 1, 2, 10]$ is the mean of $[(3-4.6)^2, (7-4.6)^2, (1-4.6)^2, (2-4.6)^2, (10-4.6)^2]$.

   Define a function `variance` that takes a non-empty list of numbers and returns its variance. Be sure to use any previously-defined relevant function.

6. (5 points) The *standard deviation* of a collection of numbers is the square root of the variance of those numbers.

   Define a function `stdDeviation` that takes a non-empty list of numbers and returns its standard deviation. Be sure the to use any previously-defined relevant function.

Predator/prey (30 points total)

7. (30 points – 6 points each) Consider the predator/prey code developed in class (Wednesday 18 January). For each of the following behaviors, either find example numbers such that the populations returned by `populations(...)` and plotted by `plotPopulations(...)` illustrate the behavior and explain why the behavior is reasonable for that set of inputs, or describe why that behavior is not possible.

   (a) Both the hare and lynx populations oscillate.

   (b) The hare population continually increases.

   (c) The lynx population continually increases.

   (d) The hare population continually decreases.

   (e) The lynx population continually decreases.

For COMP 130 students

Sum of this and sum of that (10 points total)

8. (10 points) In the "finger exercises", we saw a simple built-in function `sum()` to sum all the numbers of a list. We would now like a more flexible version of this. Given three arguments, a list of numbers, a starting index, and an ending index, we want a function `sumList` that returns the sum the list elements from the starting index, up to but not including the ending index. E.g., `sumList([5,9,2,-3,8,1], 2, 4)` would be `-1`, since $2 + -3 = -1$. You are not expected to handle invalid index values.

   Define this function.

More Statistics (40 points total)

9. (15 points) The *lower median* of a collection of numbers is, intuitively, the "middle" number when looking at the numbers in order. E.g., the lower median of $[5, 2, 3]$ is 3, and the lower median of $[5, 2, 8, 3]$ is also 3 (and not 5). The "lower" part refers to which of the middle two numbers is the result when we have an even-sized collection.

   When we have i numbers, the lower median is the j-th smallest of these, as described by the following table:

i	j
1	1
2	1
3	2
4	2
...	
17	9
18	9
...	

   The median is often used as a substitute for the better-known arithmetic mean, as it is less affected by having a few extreme values.

   Define the function `lowerMedian` that takes a non-empty list of numbers and returns its lower median. Be sure the to use any previously-defined relevant function.

10. (25 points) The *simple moving average* of a series of numbers is, intuitively, a series of averages of some of those numbers. Given a series of numbers a_i and a number n, the moving average is the series containing the average of a_1 through a_n, the average of a_2 through a_{n+1}, the average of a_3 through a_{n+2}, etc. Note that the average of a_1 through a_m, where $m < n$ is not included, so the resulting series is shorter than the input. For example, the simple moving average of $[5, 2, 8, 3, 7, 4]$ for $n = 3$ is $[(5 + 2 + 8)/3, (2 + 8 + 3)/3, (8 + 3 + 7)/3, (3 + 7 + 4)/3]$.

    The simple moving average is often used is investment analysis and scientific data analysis to smooth out fluctuations in the data. The following illustrates three different moving averages, for different values of n:

    Define the function `SMA` that takes a list of numbers and a positive integer (n) and returns a list of numbers that are the appropriate averages. Be sure the to use any previously-defined relevant function.

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?