Week 15, Lecture 1

We have completed the course syllabus and your final test is scheduled for Thursday, December 1. It is supposed to be comprehensive but will focus on the material we have covered since Test 2.

In these last four lectures, we will cover material from the book that we have not had a chance to cover so far. To help with that, please read the slides for Chapter 10 from the text and , in particular, (a) the cannonball example, and (b) the many-sided die example. Both examples are straightforward, and once you understand them, you'll see how they can be set up as classes so that you can construct cannonball objects and many-sided die objects just as easily as you obtained integers, strings and lists ..... only this time, the objects are ones for which you have written the classes yourself.

So what is an object? Think of it as "something that has (a) some data and thus knows this data about itself, and (b) a number of functions (called methods) that can operate on this data."

You'll notice that using objects makes your programs clean and modular. Your programs start to become self-contained (encapsulated) and simultaneously more general.

To begin with, read the hand-written pdf describing the

cannonball problem. Supplement this with the slides from Chapter 10 and make sure you understand the cannonball problem and the solution method. This is called an approximate, iterative solution. It is approximate because we make assumptions (e.g., no wind resistance) and because we iterate over small time-steps using an approximation for velocity in the y-direction.

Thus, with an approximation in the x-direction (no wind resistance) and an approximation in the y-direction (i.e., though the y-velocity changes continuously, we use an average velocity over the time step based on only the start and end velocities over that time step), the entire solution becomes an approximation.

It is iterative because we are computing the position of the cannonball over consecutive time-steps. We cannot get its position at some arbitrary time without going step by step. You could solve the problem accurately using calculus, but that requires a little more math than we use here.

The first program involves an iteration for the cannonball problem.

The second file (not a program) demonstrates how the look of the program changes when you take an object-oriented view.

The third program shows you how to write a class to construct many-sided die objects and then use such

objects in a simple game. Once you have such a class you can use these objects anywhere.

The fourth program builds upon what you just saw in the third program to develop a class for the cannonball program.

The fifth program puts some very trivial graphics in the cannonball code. You can improve on these graphics as an exercise.