PHY 250L – Spring 2018
Programming assignment 0

Hello, and welcome to the PHY 250L programming tutorial. This week, we began exploring the wonders of python! The goal of the next few weeks of the lab is to introduce you to a tool of great utility in scientific research: scientific computing. We'll be investigating ways to use computing for calculations, visualizations, and simulations.

Things you should understand after week 0:

- Jupyter notebook
- code cells, markdown cells
- "SHENTER"
- variables
- mathematical syntax
- printing to screen
- control structures/loops: for, while, if, unless

Problems for 2.15.2018 The following problems should be completed and uploaded to Sakai by 09:45 on 2.8.2018. PLEASE create a subdirectory for this assignment. The solutions to all problems should be contained within a single Jupyter notebook. Each problem should have its own cell(s), and the problems should be separated by headings in markup cells. Please don't just glom your problems onto the end of the tutorial file; start a new file.

1. `fibonacci1`, 30 points

   Write a program that computes the first 2×10^3 numbers in the Fibonacci sequence. What's *that*, you ask? Oh, I'm so glad that you asked.

   The n^{th} Fibonacci number, f_n, is given by

   $$f_n = f_{n-2} + f_{n-1}, \tag{1}$$

   with the specification that $f_0 = 0$ and $f_1 = 1$. The Fibonacci sequence is the ordered collection of all such numbers. Eq. 1 is an example of what is called a *recursion relation*, since it relates a value in a sequence to the values that come before it. We sometimes call f_0 and f_1 the *seed* values of the sequence, from which the rest of the Fibonacci numbers grow like so many flowers in a garden.

   Your program should compute up to f_{2000}, and write to screen three columns of numbers:

   $$i \qquad f_i \qquad f_i/f_{i-1} \tag{2}$$

   Columns should be separated by a tab character, "`\t`".

   Western study of the Fibonacci sequence dates back to the Pisan Leonardo Fibonacci, who published it as a method for modeling population growth in 1202 AD. The concept had been studied in the east, particularly in India for at least 10^3 years prior to Fibonacci! The sequence, and the more general recursion principle on which it is based, appear in many places in the natural world. Have a google.

Does the ratio that you calculate trend toward a single value?

2. `oneprime`, 10 points
   Write a program that finds the factor of the number 3462457544 Recall that p is a factor of n if $n \cdot m = p$ for some integer m. Your program should output each factor to screen as you find it.

3. `primes100`, 30 points
   Write a program that outputs the first 100 prime numbers. Your program should perform factorization test on candidates. Bonus points if your program will also output the time it takes to run.

4. `ducks`, 30 points
   A lake has 2500.0 ducks on it in year 0. Each year, the duck population increases by 6.2%, but if the population goes over 3201, the lake gets too crowded and duck flu kills off 1428 ducks. If the number of ducks is below 2500, there is abundant food and the number of ducks increases by 8.9% annually. Find the number of ducks on the lake in the year 3000. Your program should print to screen the duck population each year.

   Big hint (Possible life advice, too. Works on many levels.)
   The types of problems that we solve with computers are typically really tricky. If they weren't, we'd just grind them out with pen and paper! In cases where problems are really complicated, it's a good idea to break them down and solve piece-by-piece. Knowing how to do this is often what makes a researcher good. So how do you do it? PRACTICE.

   In the duck case above there are several steps. Here's how I see it:

   - First, make the program begin with 2500 ducks.
   - Then, make the number of ducks increase by 6.2% each year for 3000 years.
   - Then, make the program spit out the number of ducks after 3000 years.
   - Then, make the program check if the number is above 3201.
   - Then, make the program kill a bunch of ducks if the number is above 3201.
   - Then, make sure the number keeps increasing after all thos ducks die.
   - Then, make the program check if the number of ducks is below 2500.
   - Then, if the number of ducks is below 2500, make the growth rate 8.9%.

   In implementing this, it's almost always a good idea to have the program print information at each step. Make sure the program is behaving as you expect!