Lab Projects for July 11, 2012

These projects will get you programming in Python. They range from the straightforward (where we give you the formulae and you have to program them) to a couple that require you to think about how to do them

Compound Interest

We want to know how much money F will be available in Y years if we invest the principle P at rate r, compounded n times each year. The formula is:

$$F = P\left(1 + \frac{r}{n}\right)^{nY}$$

1. Write a program to compute the amount that \$1000 will produce if invested at 8% compounded semiannually every year for 5 years. Your output is to look exactly like this:

```
The amount that $1000 produces when invested at 8%
and compounded 2 times a year:
 1 $1081.60
 2 $1169.86
 3 $1265.32
 4 $1368.57
 5 $1480.24
```

2. Do the same, except compound the amount daily (that is, 365 times per year).

3. Now prompt the user for the principle, rate (as a percent), and number of times compounded per year. Then compute how much money will be available, just as in part 1, and print it in the same format (although with the appropriate number of lines of output).

Collatz Conjecture

Define the function:

$$f(n) = \begin{cases} n/2 & \text{if } n \text{ is even} \\ 3n+1 & \text{if } n \text{ is odd} \end{cases}$$

The *Collatz conjecture* says that, if you iterate this sequence for any initial value of n, then eventually the sequence will reach the number 1.

For a given number n, let k be the *least* number of iterations needed to reach the number 1 (excluding the initial value). Then k is called the *total stopping time* of n.

For example, if $n = 29$, then the sequence is:

```
29 88 44 22 11 34 17 52 26 13 40 20 10 5 16 8 4 2 1
```

and so the total stopping time of 29 is 18.

Write a program that takes as input a positive integer and prints both the sequence and the total stopping time for that integer. The output should look like:

```
29 88 44 22 11 34 17 52 26 13 40 20 10 5 16 8 4 2 1
The total stopping time for 29 is 18
```

The Smallest Number

When you're dealing with floating-point numbers, "precision" is the number of digits to the right of the decimal point.

Every computer is limited in the amount of precision it can represent for floating-point numbers. Let's let ϵ be a variable that holds floating-point numbers. At some point, when the value stored in ϵ is very small, the following expression will be true:

$$1.0 == 1.0 + \epsilon$$

In other words, the value of ϵ becomes so small compared to 1.0 that the computer ignores it, in essence.

Also, computers have at least two ways to represent floating-point numbers: a single-precision floating point number and a double-precision floating point number (which basically doubles the precision). For example, in the C programming language, these are referred to as `float` and `double` respectively.

Write a program to find the largest value of ϵ in the above expression on your computer. Also, what happens if you try to do this but use the expression

$$0.0 == 0.0 + \epsilon$$

Computing π

Write a program that approximates the value of π by summing the terms of this series:

$$\pi = \sum_{i=0}^{\infty} \frac{4(-1)^i}{2i+1} = \frac{4}{1} - \frac{4}{3} + \frac{4}{5} - \frac{4}{7} + \frac{4}{9} - \frac{4}{11} + \ldots$$

Prompt the user for n, the number of terms to sum, and then output the sum of the first n terms of this series. Your program is also to print the absolute value of the difference between the approximation and the value of `math.pi` to see how accurate the approximation is.

Don't forget to put `import math` at the top of your program!

The Monty Hall Problem

Suppose you're on a game show (well, it was called *Let's Make a Deal*). You are given the choice of three doors. Behind one door is a car; behind the others, goats. You pick a door, say Door No. 1, and the host, who knows what's behind the doors, opens another door, say Door No. 3, to show a goat. He then says to you, "Do you want to change to Door No. 2?" The question is, is it to your advantage to switch your choice?

Write a program to simulate this game show, having the contestant switch doors when asked. See whether the best strategy is to switch or to stay with the door you originally selected. Run it for 10,000 contests. What do you think the contestants should do?

Computing π Using a Monte Carlo Method

An interesting way to compute the value of π is to toss darts at a dart board. Draw a unit circle in a 2×2 square. That's your dart board. Throw a large number of darts at the board. Count the number of darts that land inside the circle, divide it by the total number you threw, and multiply by 4. If you threw enough darts, you'll get a very good approximation of π.

Here's how you do this with a computer. Consider the dart board and the circle to be centered at $(0, 0)$. That means the corners of the dart board are $(1, 1)$, $(1, -1)$, $(-1, 1)$, $(-1, -1)$, and the circle has radius 1. Generate two random[1] numbers between -1 and 1 inclusive. That's the point at which the dart landed. Then see if the point is in the unit circle. To do this, compute

$$x^2 + y^2$$

(where x is the first number and y the second). If it's less than or equal to 1, it's in the circle; if it's greater than 1, it isn't. Then count the total number of points, and the number of points in the circle. Divide the latter by the former, multiply by 4, and you have your approximation.

Write a program to do this. Try it with 100, 1000, 10,000, and 100,000 points. How good is your approximation?

[1] Pseudorandom, actually.