MCL summer workshop in graph theory
Lab 1

6/23/16

Programming exercises

Problem 1: 99 bottles

Write a program, `bottles.py`, that prints the verses of the song "99 Bottles of Beer on the Wall." Your output should look something like this:

```
99 bottles of beer on the wall!
98 bottles of beer on the wall!
⋮
1 bottle of beer on the wall!
No more bottles of beer on the wall!
```

Hints:

- Use a loop (either `for` or `while`) to go from 99 to 0.
- Notice that 0 and 1 are special cases. Use `if` statements to test for them!

Problem 2: $1 + 2 + \ldots + n$

Write a program, `sum.py`, that calculates the sum of the integers from 1 to n and prints it out. Your program should ask the user for an integer n, store it in a variable, then use a loop to add up the numbers. Hint: use another variable, `sumsofar`, to keep track of the current sum, and add a number to it in each iteration.

Sample output:

```
n = ?  5 (the gray signifies user input)
sum = 15
```

You can use the formula $1 + 2 + \ldots + n = \dfrac{n(n+1)}{2}$ to check you're work. We will prove this formula using *mathematical induction* within the next few days!

Problem 3: 1, 2, 3, 4, 5, 6, clap!

Here's another fun game to play on long car rides: go around in a circle, counting 1, 2, ..., but whenever you get to a *multiple of 7* or a *number with 7 as a digit*, clap instead of saying the number. Go until someone messes up, then start over.

1. For simplicity, let's forget about the numbers with 7 as a digit. Write a program that counts from 1 to 100, replacing the multiples of 7 with the word "clap." Hint: use the `%` operator to test for multiples of 7.

2. Let's say that on every "clap" we have a 50% chance of messing up and saying the number instead. Modify your program to incorporate this behavior. Hint: you can use the function `random.randint(0, 1)` to simulate a coin flip.

3. Modify your program so that, instead of always stopping at 100, it stops at the first mess up. Your program now has the potential to run forever, but it's very unlikely that it will do so. Still, if it goes on too long you can press `Ctr-C` to stop it.

Extra credit: try to make your program clap on numbers with 7 as one of the digits.

Problem 3: Func-ify it!

If problems 1-3 were too easy for you CS nerds, the next thing we'll be covering is functions. Function syntax is a little different in Python versus Java:

`def` *functionName*`(`*params*`):`
    Body of function
    ⋮
    `return` *value*

Let's write functions for problems 1-3.

1. Write a function `bottles(`n`,` *beverage*`)` that recites "n Bottles of *beverage* on the Wall" instead of 99 bottles of beer. Prompt the user for a number n, store it, then call your function.

2. Write a function `sumofints(`n`)` that calculates the sum of the integers 1 through n. Your function should not print anything, but should *return* the value of the sum. Outside the function, prompt the user for n, store it, call your function, and print the answer.

3. Write a function `clapgame(`q`)` which plays the clapping game with multiples of q instead of 7.