COSC 120 / Martin / Fall 2014 Assignment #1
Computing primes and prime factorizations

Assigned: Wednesday, September 10th
Due: Thursday, September 18th, 11:55pm

Introduction

In this assignment you will write two small programs involving prime numbers. These programs will require the use of conditional statements and loops.

Collaboration

You may work with other students, but each of you must submit your own assignment. If you work with others, list their names at the top of your program as you did for the previous assignment. Please read the collaboration policy in the course syllabus.

Problem #1: Finding primes

Your first program will print a list of the first N prime numbers, where N is a number entered by the user. Recall that a number is prime if its only divisors are 1 and itself. For technical reasons, the number 1 is not considered prime (ask your local mathematician), so the first few primes are 2, 3, 5, 7, and 11.

Write a program that does the following:

1. Asks the user how many primes should be printed. (Call this number N.)
2. Prints a list of the first N primes.

Save your program as `prob1.py`.

Your program should behave as shown in the sample run below. User input is shown here in bold for clarity and several hundred lines of output have been omitted to save space.

```
Number of primes to print? 1000
Prime #1 is 2
Prime #2 is 3
Prime #3 is 5
Prime #4 is 7
Prime #5 is 11
...
Prime #999 is 7907
Prime #1000 is 7919
```

Hints:

- When checking whether some number n is prime, it is certainly sufficient to check that no number from 2 up through $n-1$ is a divisor of n. If you give it some thought, however, you can convince yourself that you don't need to check all the way up to $n-1$ (and hence your program will run a bit faster). When can you stop?

- A variable to which you assign a boolean value (True or False) can be a useful way to keep track of whether some condition has occurred or not.
- Remember that a break statement can be used to immediately exit the nearest enclosing while or for loop in which it occurs.

Problem #2: Finding the prime factorization of a number

The second problem asks you to write a program that finds the prime factorization of a number entered by the user. Every number (2 or greater) can be factored into primes. For example, 35=5*7,125=5*5*5=53, and 600=(2*2*2)*3*(5*5) = (2**3)*(3**1)*(5**2).

Write a program that does the following:

1. Asks the user to enter a number greater than or equal to 2.
2. Prints a list of the prime factors of the user's number, with their corresponding exponents, as shown below.

Save your program as prob2.py.

Your program should behave as shown in the sample runs below. User input is shown here in bold for clarity.

```
Enter a number (must be 2 or greater): 35
The prime factorization of 35 is:
 p   exponent
 - - - - - - - - -
 5   1
 7   1

Enter a number (must be 2 or greater):600
The prime factorization of 600 is:
 p   exponent
 - - - - - - - - -
 2   3
 3   1
 5   2
```

Hints:

Below is a rough outline of one possible approach. Please feel free to use it, modify it, or ignore it completely.

1. Prompt the user to enter a number; store the number in a variable.

2. Print the first part of the output, up to and including the table headings and the dotted line.

3. Initialize some variables that will be used to keep track of things such as the current divisor being tested and the exponent for that divisor.

4. For each possible divisor (2, 3, 4, ... up to some appropriate stopping point) of the user's number:

   (a) check whether it is prime (you should be able to reuse some of your code from Problem 1 for this, after making appropriate modifications).

   (b) If it is prime and it is a divisor of the user's number:

      i. Determine the highest power of it that is a divisor. (One interesting approach to this: Suppose the user's number is 72, for example, and the current prime divisor you're checking is 2. Is 2 is a divisor of 72? Yes, and 72/2 is 36. Is 2 a divisor of 36? Yes, and 36/2 is 18. Is 2 a divisor of 18? Yes, and 18/2 is 9. Is 2 a divisor of 9? No. How many iterations did we go through before we got an answer of "No"? Three. So 2^3 is the highest power of 2 that is a divisor of 72.)

      ii. Print one line of the table of results.

   (c) Update the values of variables as necessary. Go on to the next possible divisor. Notice that this approach requires an outer loop that generates the possible divisors one at a time, with two loops inside of it, one to check whether something is prime and one to find the exponent for a prime divisor. Toward the end of each iteration of the outer loop, one line of the table of results might be printed.

Submission Procedure

At the top of each of your programs (prob1.py and prob2.py), in a comment, list the names of any students with whom you collaborated, and give the approximate number of hours you spent on this assignment. For example:

```
# COSC 120 Assignment 1 Problem 1
# Name: Stimpson J. Cat
# Collaborators: Ren Hoek, Secret Squirrel
# Time spent: 0:45 ...
```

Put both files in a zip file called assign1.zip, and use the submit link to upload this file.