Lab 3: Prime Numbers – Loops

Lab Assignment for PROG1205 – Intro to Programming

Prior to attempting this problem, you should have done the following:
1. Read the assigned textbook readings for weeks 1, 2, 3, 4 and 6.
2. Viewed the Control Structures - Iteration/Loops videos from week 6 and completed the associated Pre-Class Activity.

General Requirements

1. Unlike prior lab assignments, this lab assignment is to be completed by **teams of two students**. The teams will be assigned by the instructor in class. Individual submissions will only be considered by exception, and only when negotiated in advance of the due date.
2. Analyze the problem, design a documented plan (i.e. flowchart or pseudo-code), and code and test a solution **following the step-by-step approach** presented in this course.
3. Submit your solution and plan to the appropriate drop-box in DC Connect **by the due date provided**.
4. Be prepared to **present your solution** with your partner to the instructor during a class session. This is when feedback will be provided and **grades will be assigned**.

Program Requirements

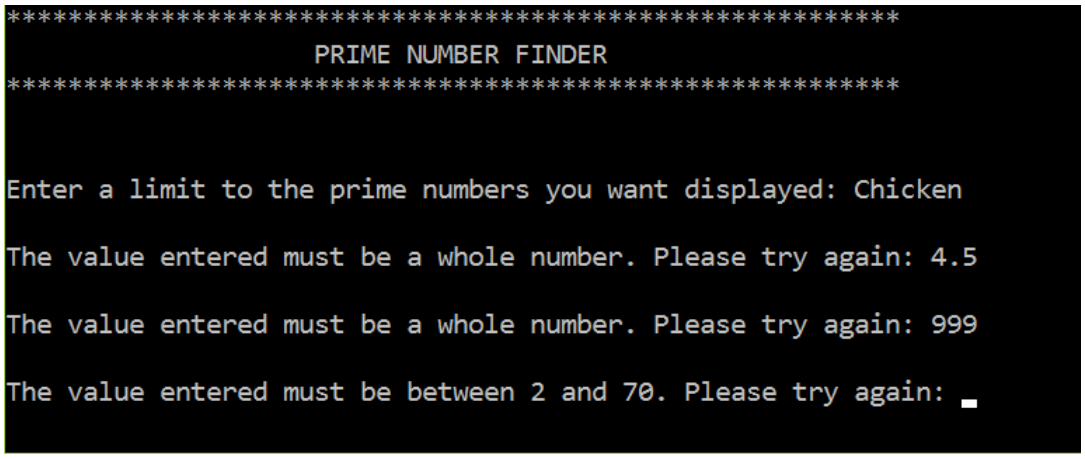
For this lab you will create a simple console application using Python that will graphically display all the prime numbers up to a limit selected by the user. For more information on prime numbers, refer to the handout "**Brief Notes on Prime Numbers**" included with these instructions. This lab will provide you with the opportunity to design program logic with both individual and nested loops. It should also reinforce the time-saving value of planning before you code.

Getting User Input:

In the input phase of the program, display a program header and then prompt the user for the limit to the prime numbers they want displayed. This input must be a whole number between the range of 2 and 70 inclusive. Should the user enter something other than a valid, in-range number, then display an appropriate message that describes the problem and give the user the opportunity to input the limit again.

You will need to use "try … except" or another means verifying whether the user input is numeric and a whole number. Also, make sure that no processing occurs until the application has valid input.

The example below shows multiple invalid inputs; match this as closely as possible:

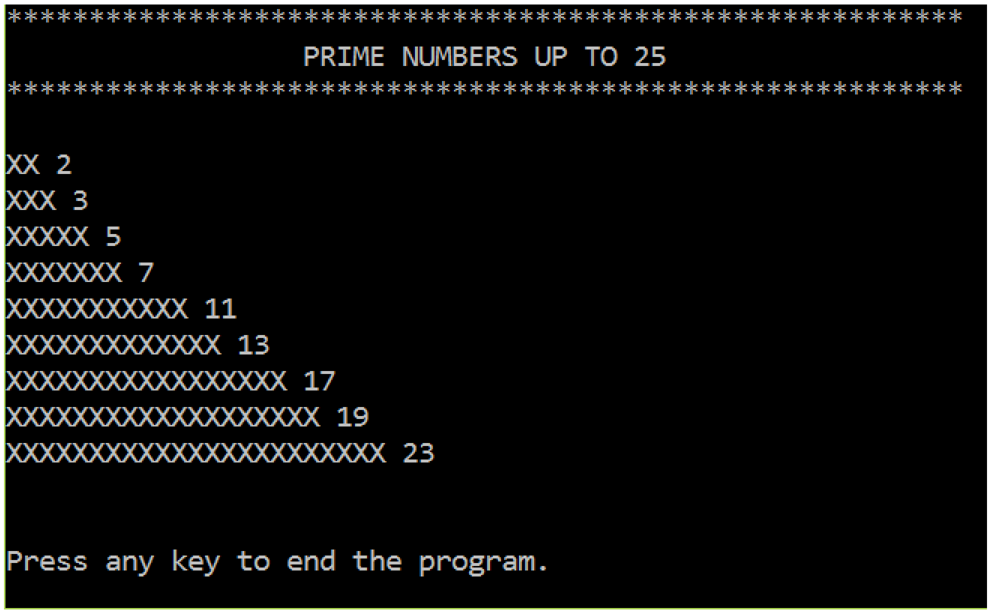


```
**********************************************************
                    PRIME NUMBER FINDER
**********************************************************


Enter a limit to the prime numbers you want displayed: Chicken

The value entered must be a whole number. Please try again: 4.5

The value entered must be a whole number. Please try again: 999

The value entered must be between 2 and 70. Please try again: _
```

Final Output:

Once you have valid input, you will produce a list of prime numbers starting at 2, going up to the limit the user entered. Start by displaying a title as similar to the example below. In addition to displaying the prime number numerically, you will also display a "bar" of 'X' characters to represent the number. For example, the prime number 2 would be represented as "XX 2".

The following output example shows what your program should produce if the user entered limit was 25; match this as closely as possible:



```
**********************************************************
                  PRIME NUMBERS UP TO 25
**********************************************************


XX 2
XXX 3
XXXXX 5
XXXXXXX 7
XXXXXXXXXXX 11
XXXXXXXXXXXXX 13
XXXXXXXXXXXXXXXXX 17
XXXXXXXXXXXXXXXXXXX 19
XXXXXXXXXXXXXXXXXXXXXXX 23


Press any key to end the program.
```

Style Guide

To be eligible for full marks on this or any lab in this course your application must conform to the requirements as outlined above as well as our prescribed style guide, in this case making sure to observe the PEP8 naming conventions for Python as well as appropriate and complete program documentation.

Development Hints:

- It is strongly recommended that you approach this lab following the "from plan to program" processes rather than try to modify a previous lab, or code it without a plan.
- Remember that the aim of this lab is to have you practice using loops. There is more than one opportunity to do so in this problem. An efficient, working solution will probably involve at least three different loops.

Brief Notes on Prime Numbers

What is a Prime Number?

A prime number is a *natural number* (positive integer) divisible by exactly two natural numbers; one and itself. The first few prime numbers are: 2, 3, 5, and 7. If you are looking for a detailed explanation of prime numbers, you can check out the video Prime Numbers by Khan Academy (8:12).

A Simple Primality Test

A simple (but not necessarily efficient) way to test if any given number is a prime number is as follows:

1. If the candidate number is not a whole number larger than 1, it is not prime.
2. If it is 2, it is prime.
3. If it is larger than 2, try dividing the candidate number by 2. If it divides evenly, it is not prime.
4. Otherwise, keep trying to divide the candidate number by increasing integers until you find a number that the candidate divides by evenly. Stop trying when the number you are dividing by is larger than the square root of the candidate number.
5. If you found a number that the candidate divides by evenly, then the candidate number is not prime. If you tested all the numbers up to and including the square root of the candidate number and none of them divided evenly, then the candidate number is prime.

Examples:

Candidate number: **47** (square root: approximately 6.9)

47 / 2 = 23.5… 47 / 3 = 15.67… 47 / 4 = 11.75… 47 / 5 = 9.4… 47 / 6 = 7.8…

Stop because 7 is greater than 6.9. No numbers found that divided evenly into 47; therefore **47 is prime**.

Candidate number: **45** (square root: approximately 6.7)

45 / 2 = 22.5, 45 / 3 = 15…

Stop because a number was found that divided evenly into 45; therefore **45 is NOT prime.**