# COP 3223 Program #4: Numbers and Designs

## Program A: K-Divisible Numbers (kdiv.py)

An k-divisible number is a number that has at least k positive integer divisors. For example, 12 is a 5-divisible number and a 6-divisible number, but it's not a 7-divisible number, because 12 has six divisors: 1, 2, 3, 4, 6 and 12.

Write a program that prompts the user to enter the three following values:

1) The value of k.
2) A lower bound, inclusive
3) An upper bound, inclusive

Your program should print out each number in the desired range that is a k-divisible number.

## Input Specification

1. k will be a positive integer in between 2 and 20, inclusive.
2. The lower bound will be a positive integer in between 1 and 1000, inclusive.
3. The upper bound will be a positive integer in between the lower bound and 1000, inclusive.

***Note: YOU DO NOT HAVE TO CHECK TO SEE IF THESE RESTRICTIONS ARE MET WITH IF STATEMENTS IN YOUR CODE. RATHER, THIS IS A GUARANTEE THAT THE GRADER WILL NOT USE TEST CASES OUTSIDE OF THESE BOUNDS.***

## Sample Run #1 (User input in bold and italics)

```
What is k?
6
What is your lower bound?
10
What is your upper bound?
30
Here are all of the 6-divisible numbers from 10 to 30:
12 18 20 24 28 30
```

## Sample Run #2 (User input in bold and italics)

```
What is k?
15
What is your lower bound?
100
What is your upper bound?
180
Here are all of the 15-divisible numbers from 100 to 180:
120 144 168 180
```

**Note: Use end and sep appropriately to get output similar to that shown above.**

**Program B: Stars, no Stripes (stars.py)**

Write a program that prints out the stars for a flag design like the American Flag, but where the user enters the number of rows and the user enters the number of stars on the first row. The second row has one fewer star than the first and every other row is identical. Copy the spacing shown in the examples. **Note: To get full credit, you MUST use a nested loop structure and you can NOT multiply a string by a number to print out that string multiple times. This is because the goal of this assignment is to practice nested loops.**

**Input Specification**

1. The number of rows will be a positive integer in between 1 and 20, inclusive.
2. The number of stars in the first row will be a positive integer in between 2 and 30, inclusive.

**Sample Run #1 (User input in bold and italics)**

```
How many rows for this design?
9
How many stars on the first row?
6
* * * * * *
 * * * * *
* * * * * *
 * * * * *
* * * * * *
 * * * * *
* * * * * *
 * * * * *
* * * * * *
```

**Sample Run #2 (User input in bold and italics)**

```
How many rows for this design?
2
How many stars on the first row?
30
* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
```

**Program C: Cube (cube.py) or Nested Squares (squares.py)**

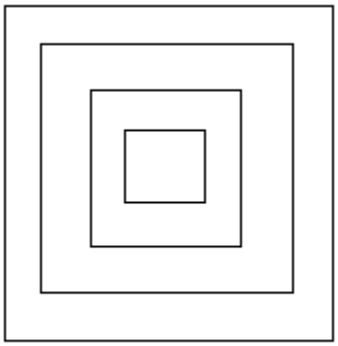
For this program you will use the python turtle and you have a choice of what you want to draw. If you choose to draw a cube in perspective, then name your file cube.py. If you choose to draw nested squares, then name your file squares.py.

**Option 1: Cube (cube.py)**

Cubes drawn on paper are created using a technique called perspective. In this technique, line segments that are parallel in actuality are not necessarily parallel on paper. This helps others viewing the drawing perceive depth. Using trial and error, use the python turtle to draw a cube using the perspective technique. You will be graded on the "relative" positioning of your lines and can get a perfect score without the drawing being accurate by art standards.

**Option 2: Nested Squares (squares.py)**

Here is a picture of four (somewhat) symmetrically nested squares:



(I drew this in MS Paint, so it's far from perfect.)

For this program, ask the user for the number of squares to enter as well as the x coordinate of the top right corner of the smallest square. Center your drawing at (0, 0) and make the x coordinate of the top right corner of the second smallest square to be twice that of the smallest square. Make the x coordinate of the top right corner of the third smallest square to be three times of that of the smallest square, and so forth.

We will only test your program on values that safely fit on our screen. If the user enters 5 squares with an x coordinate of 10 for the top right corner of the smallest square, then the corners of the five squares are:

1) (10, 10), (10, -10), (-10, -10), (-10, 10)
2) (20, 20), (20, -20), (-20, -20), (-20, 20)
3) (30, 30), (30, -30), (-30, -30), (-30, 30)
4) (40, 40), (40, -40), (-40, -40), (-40, 40)
5) (50, 50), (50, -50), (-50, -50), (-50, 50)

**Beginning of a Sample Run (corresponding to description above)**

```
How many squares for this design?
5
What's the x coordinate of top right corner of the first square?
10
```

**Deliverables**

Please submit three separate .py files for your solutions to the following problems via WebCourses by the designated due date:

Program A: **kdiv.py**
Program B: **stars.py**
Program C: **cube.py OR squares.py (not both, only one)**