CMSC 201 Spring 2017
Project 1 – Number Classifier

Assignment: Project 1 – Number Classifier
Due Date:
Design Document: **Saturday**, March 11th, 2017 by 8:59:59 PM
Project: Friday, March 17th, 2017 by 8:59:59 PM
Value: 80 points

Collaboration: For Project 1, **collaboration is not allowed** – you must work individually. You may still come to office hours for help, but you may not work with any other CMSC 201 students.

Make sure that you have a complete file header comment at the top of each file, and that all of the information is correctly filled out.

```
# File:     FILENAME.py
# Author:   YOUR NAME
# Date:     THE DATE
# Section: YOUR DISCUSSION SECTION NUMBER
# E-mail:   YOUR_EMAIL@umbc.edu
# Description:
#    DESCRIPTION OF WHAT THE PROGRAM DOES
```

Project 1 is the first assignment where you've had to turn in a "design document" in addition to the actual code. The design document is intended to help you practice deliberately constructing your program and how it will work, rather than coding as you go along, or starting without a plan.

Instructions

For this project, you will be creating a single program, but one that is bigger in size and complexity than any individual homework problem. This assignment will focus on using functions to break a large task down into smaller parts.

The design for Project 1 will be explained *in class*, and will be posted to the course Blackboard after the last section meets on Tuesday.
Do not start coding Project 1 until you have seen the design overview!

You are ***required** to follow the provided design exactly*! You may add additional functions, but you must implement all of the specified functions as described in the design overview.

At the end, your Project 1 file must run without any errors.
It must also be called proj1.py (case sensitive).

Additional Instructions – Creating the proj1 Directory

During the semester, you'll want to keep your different Python programs organized, organizing them in appropriately named folders (also known as directories).

You should create a directory in which to store your Project 1 files. We recommend calling it `proj1`, and creating it inside a newly-created directory called `Projects` inside the `201` directory.

If you need help on how to do this, refer back to the detailed instructions in Homework 1.

Objective

Project 1 is designed to give you lots (and lots) of practice with functions, as well as teach you some interesting things about different numbers. You'll need to use **`while`** loops, control statements like **`if/else`**, passing in parameters, returning from functions, concatenation needed for using **`input()`**, and algorithmic thinking.

Remember to enable Python 3 before running and testing your code:

```
scl enable python33 bash
```

Task

Mathematics has many subfields, but arguably one of the most interesting is number theory, which deals with integers and their relationships to one another. This field has led to discoveries such as:

- Prime numbers
- Perfect numbers
- Square numbers
- Triangular numbers

For this project, you'll be creating a program that prints out interesting facts (like those above) about a range of integers, selected by the user. The user will select a beginning number and an ending number (*e.g.*, 5 to 77).

Your program will then print out, for each number, if they are odd or even, prime, perfect, a square, or a triangular number. The output will be presented in a well-formatted table, with each number and its properties being shown in order on individual lines. (The table's columns must line up across the different lines, or the table will be unreadable.)

Specification

Prior to this assignment, you should be familiar with the Coding Standards, available on Blackboard under "Assignments" and linked on the course website at the top of the "Assignments" page.

For now, you should pay special attention to the sections about:

- Naming Conventions
- Use of Whitespace
- Comments (Header and In-Line)
    - ***Function Header Comments*** will be required for Project 1
    - Please note that the "Input" and "Output" in the function header comment do NOT mean what is shown on the screen with **`print()`**, or what is gotten from the user with **`input()`**. They refer to the **parameters** taken in, and the **return value**. (Both "Input" and/or "Output" may be none if appropriate.)
- Constants
- Make sure to **read the last page of the Coding Standards document**, which prohibits the use of certain tools and Python keywords

You should start forming good habits now. Make sure to pay attention to your TA's feedback when you receive your Homework 4 grade back, and to update your Project 1 file if necessary. If you have questions about commenting, whitespace, or any other coding standards, please come to office hours.

Additional Specifications

For this assignment, **you must follow the design overview** discussed in class (and posted on Blackboard afterwards).

For this assignment, you do need to worry about "input validation." You may assume that the user will enter an integer, but it may be negative or outside of the allowable range.
If the user enters a different type of data than what you asked for, your program may crash. This is acceptable.

Prime Numbers

Prime numbers are those numbers divisible only by one and themselves. If a number is not prime, it is called a composite number. (The number one is actually neither prime nor composite!) You can find more information, and some methods for checking "prime-ness" on the Wikipedia page.

Prime numbers have some really interesting properties. For example, two is the only even prime (because every other even number is divisible by it!). And every composite number can be expressed as a product of two or more primes. Prime numbers are also super duper important for things like public-key cryptography, which uses *really* large primes to encode sensitive information (like your password, credit card number, or personal emails).

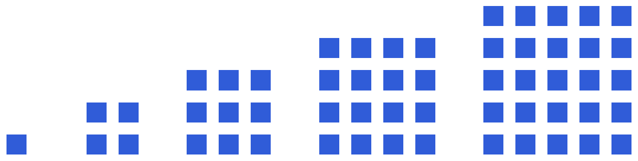
Perfect Numbers

Perfect numbers are numbers that are equal to the sum of their divisors. For example, 6 is divisible by 1, 2, and 3. Since 1 + 2 + 3 = 6, that means that 6 is a perfect number. The next perfect number is 28 (1 + 2 + 4 + 7 + 14), followed by 496, and then 8128. The next one isn't for a while – 33550336, to be exact. Numbers that aren't perfect are either abundant (the sum of their divisors is larger than them) or deficient (the sum of their divisors is smaller than them). You can find more information on the Wikipedia page.

Perfect numbers have some really interesting properties, especially in relation to binary numbers. (Binary is another way of representing numbers, and we'll learn it later in the semester!) Additionally, all known perfect numbers end in 6 or 8; there are no known odd perfect numbers.

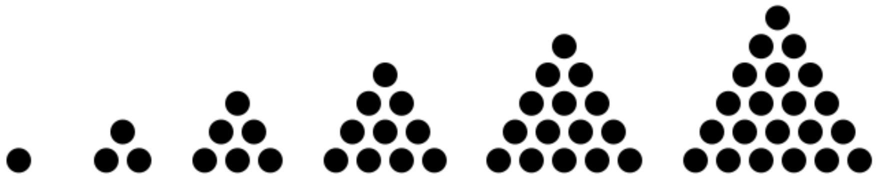
Square Numbers

You should be familiar with square numbers from high school math – 1, 4, 9, 16, 25, 36, 49, 64, etc. These are all "perfect" squares, because they can be expressed as one integer multiplied by itself. They are called "square numbers" because a group of 1, 4, 9, 16, etc. dots can be arranged in a square. See the Wikipedia page for more information.


(image adapted from https://en.wikipedia.org/wiki/Square_number)

Triangular Numbers

Triangular numbers are those numbers that, if we stacked objects, would form a perfect equilateral triangle. (Think of the ten pins in bowling, which form an equilateral triangle when they're arranged on the lane.) All triangular numbers are the sum of consecutive integers: 1 + 2 = 3; 1 + 2 + 3 = 6; 1 + 2 + 3 + 4 = 10 and so on. For more info, see the Wikipedia page.


(image adapted from https://en.wikipedia.org/wiki/Triangular_number)

A natural example of a triangular number occurs when a group of people meet for the first time, and everyone shakes hands with everyone else. It shows up similarly when sports teams have playoffs: a group of 4 teams needs 6 matches, a group of 5 teams needs 10, and 8 teams need 28.

Project

The project is worth a total of 80 points. Of those points 10 will be based on your design document, 10 will be based on following the coding standards, and the other 60 will be based on the functionality and completeness of your project.

Design Document

The design document will make sure that you begin thinking about your project in a serious way early. This will not only give you important experience doing design work, but will help you gauge the number of hours you'll need to set aside to be able to complete the project. **Your design document must be called design1.txt.**

For Project 1, we have discussed the design overview in class, and you are **required to follow it**. For future projects, you will be creating the design entirely on your own, and may choose to design it however you like.

Your design document must have four separate parts:

1. A file header, similar to those for your assignments
2. Constants
    a. A list of all the constants your program will need, including a short comment describing what each "group" of constants is for
3. Function headers
    a. A complete function header comment, including the description, inputs, and outputs
4. Pseudocode for main()
    a. A brief but descriptive breakdown of the steps your main() function will take to completely solve the problem; note function calls under the relevant comment (if applicable)

Although you will be presented with a design overview, you must still create the function headers and pseudocode for **`main()`** on your own.

A start for your design is provided on Blackboard under "Assignments". Follow the layout and format of that document. You can also copy it using:

```
cp /afs/umbc.edu/users/k/k/k38/pub/cs201/design1.txt .
```

Your **`design1.txt`** file will be compared to the **`proj1.py`** file that you submit. Minor changes to the design are allowed. A minor change might be the addition of another function, or a small change to **`main()`**.

Major changes between the design and your project will lose you points. This would indicate that you didn't give sufficient thought to your design.
(If your submitted design doesn't work, it is generally better to lose the points on the design, and to have a functional program, rather than turning in a broken program that follows the design. The ultimate decision is up to you.)

To submit your design document, use

```
linux1[4]% submit cs201 PROJ1_DESIGN design1.txt
Submitting design1.txt...OK
linux1[5]% 
```

Sample Output

The sample output is available as a separate file under "Assignments" on Blackboard, and is called "sample1.txt". Look at the sample output before reading the notes below.
(Yours does not have to match the sample output exactly, but it should be similar, and the columns should all line up nicely.)

Other Requirements

- Notice that in the primes column, the number 1 has the special case, 'Neither'. How you decide to handle this is up to you.
- The program makes it impossible to put in an invalid range (like 50 down to 10). To do this, the second time we prompt the user for an integer, the lowest number the program will let them pick is the beginning value. (You can see this in the second sample output.)
- The table should be generated by the results returned from the required functions and should work for any range within 1 to 100000, inclusive.
- You'll need to make use of the tab character, *a "`\t`" (backslash T)*, to get your first column to line up correctly. You can use either tab (or a lot of spaces) to get the rest of the columns to line up.

Submitting

Once your **`proj1.py`** or **`design1.txt`** file is complete, it is time to turn it in with the **`submit`** command. (You may also turn the project in multiple times, as you reach new milestones. To do so, run **`submit`** as normal.)

To submit your design file (which is due **Saturday**, March 11th, 2017 by 8:59:59 PM), use the command:

```
linux1[4]% submit cs201 PROJ1_DESIGN design1.txt
Submitting design1.txt...OK
linux1[5]%
```

To submit your project file (which is due **Friday**, March 17th, 2017 by 8:59:59 PM), use the command:

```
linux1[4]% submit cs201 PROJ1 proj1.py
Submitting proj1.py...OK
linux1[5]%
```

If you don't get a confirmation like the one above, check that you have not made any typos or errors in the command.

You can check that your homework was submitted by following the directions in Homework 0. Double-check that you submitted your homework correctly, since **an empty file will result in a grade of zero for this assignment.**