COP 2930 - Individual Programming Assignment #4

Due date: Please consult WebCourses for your due date/time

Objectives

1. To learn how to design a for loop.
2. To practice incorporating old information (assignment statement, arithmetic expressions, if statement) into new constructs.

Problem A: Mile Tracker A (milesa.py)

A simple running plan is to run some number of days in a row and run the same number of miles each day. For this program, ask the user how many days they want to run and how many miles a day they are going to run, and print out a summary of each day which lists the total number of miles they've run up until that day (including that day.)

Input Specification

Note: It is guaranteed that whoever uses your program will adhere to these specifications. This means that you do NOT have to check for them!

The number of days to run will be an integer in between 1 and 100, inclusive.
The number of miles to run will be an integer in between 1 and 10, inclusive.

Output Specification

For each day, output a line of the following format:

```
After day D, you will have completed M miles.
```

where D is the day number, starting at 1, and M is the total number of miles run by that day, including that day.

Output Sample

Below is one sample output of running the program. **Note that this sample is NOT a comprehensive test.** You should test your program with different data than is shown here based on the specifications given above. In the sample run below, for clarity and ease of reading, the user input is given in *italics* while the program output is in **bold**. (Note: When you actually run your program no bold or italics should appear at all. These are simply used in this description for clarity's sake.)

Sample Run #1

How many days do you plan to run?
3
How many miles per day do you plan to run?
4
After day 1, you will have completed 4 miles.
After day 2, you will have completed 8 miles.
After day 3, you will have completed 12 miles.

Problem B: Mile Tracker B (milesb.py)

Most people, on the weekend, due to having extra time, can run more. For this problem, we define the weekend to be days 6 and 7, 13 and 14, 20 and 21, etc. (Basically, count by 7s from 6 and 7.) Adjust the program from part A so you ask the user how many miles they run during the week and how many they run during the weekend, and print out a similar summary.

Input Specification

Note: It is guaranteed that whoever uses your program will adhere to these specifications. This means that you do NOT have to check for them!

The number of days to run will be an integer in between 1 and 100, inclusive.
The number of miles to run during the weekday will be an integer in between 1 and 10, inclusive.
The number of miles to run during a weekend day will be an integer in between 1 and 10, inclusive.

Output Specification

Output a statement for each day in the same exact form as program A.

Output Sample

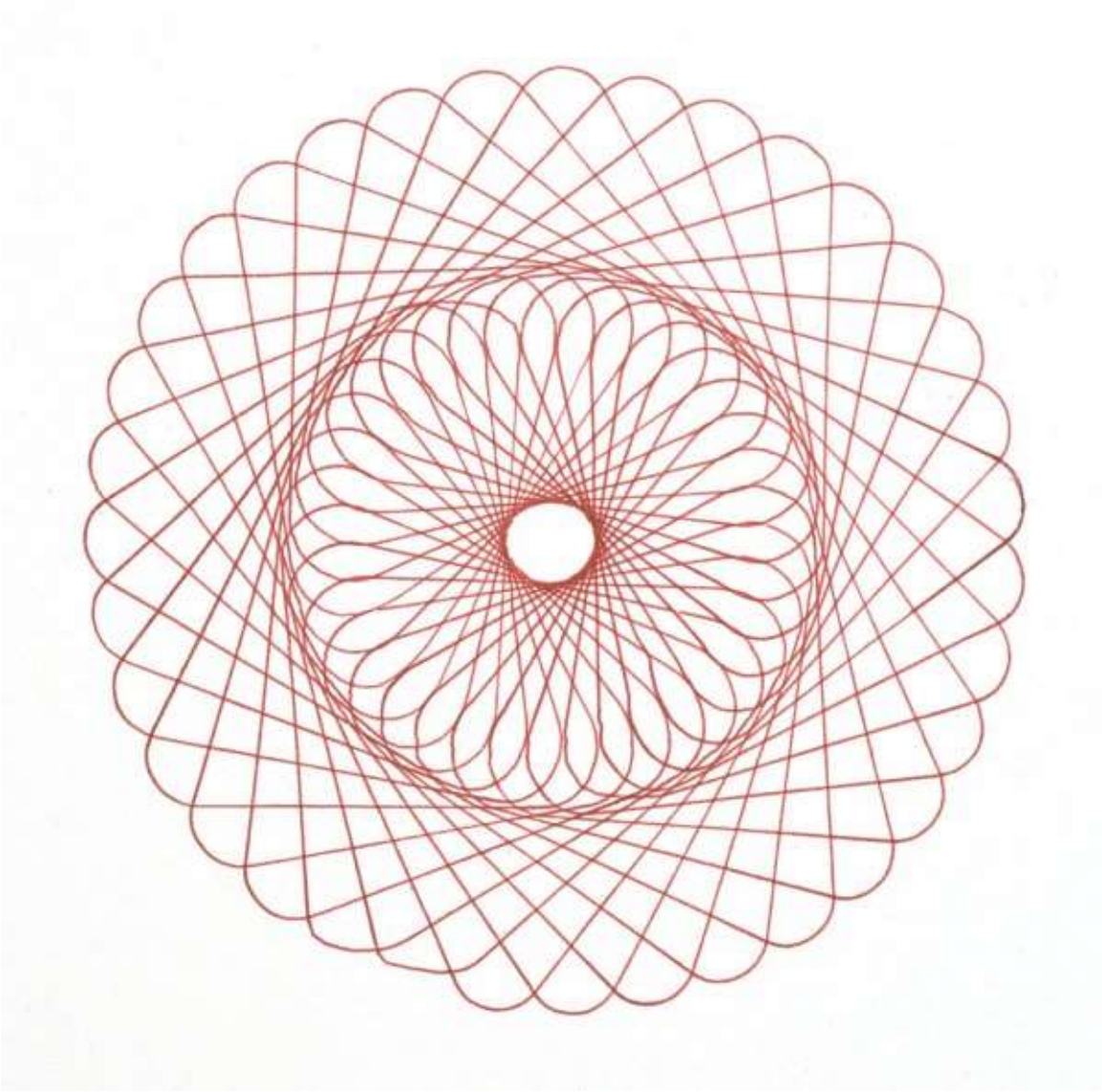
Below is one sample output of running the program. **Note that this sample is NOT a comprehensive test.** You should test your program with different data than is shown here based on the specifications given above. In the sample run below, for clarity and ease of reading, the user input is given in *italics* while the program output is in **bold**. (Note: When you actually run your program no bold or italics should appear at all. These are simply used in this description for clarity's sake.)

Sample Run #1

```
How many days do you plan to run?
9
How many miles per weekday do you plan to run?
4
How many miles per weekend day do you plan to run?
6
After day 1, you will have completed 4 miles.
After day 2, you will have completed 8 miles.
After day 3, you will have completed 12 miles.
After day 4, you will have completed 16 miles.
After day 5, you will have completed 20 miles.
After day 6, you will have completed 26 miles.
After day 7, you will have completed 32 miles.
After day 8, you will have completed 36 miles.
After day 9, you will have completed 40 miles.
```

Problem C: Python Turtle Spirograph (spirograph.py)

Repeating moving forward and turning at some specified angle can create designs similar to a spirograph, like the one shown below. Experiment with loops and drawing to create your own spirograph! (Yours doesn't need to look this cool!)



Restrictions

Please IDLE 3.6 (or higher) to develop your program. Write each in a separate file with the names specified previously, **milesa.py**, **milesb.py**, and **spirograph.py**

Each of your **three** programs should include a header comment with the following information: your name, course number, assignment title, and date. Also, make sure you include comments throughout your code describing the major steps in solving the problem.

Grading Details

Your programs will be graded upon the following criteria:

1) Your correctness

2) Your programming style and use of white space. Even if you have a plan and your program works perfectly, if your programming style is poor or your use of white space is poor, you could get 10% or 15% deducted from your grade.

3) Compatibility to IDLE.