CS 1510: Intro to Computing - Fall 2017
Assignment 4: Additive and multiplicative persistence

Due: Friday, October 5, 2017, by 11:59 p.m.

Introduction

The aim of this project is to practice the use of while loops and conditionals statements within a project that is a little bigger than the ones you have been working on. This will give you some practice with program development. Take the time to really think about this problem and work through the process before you begin to code. Write small snippets of code to make sure you can perform smaller pieces of the assignment rather than jumping in and writing the whole program at once.

Quick Overview

You are going to write a program that prompts the user for an integer and then determines the additive persistence and corresponding additive digital root, and the multiplicative persistence, the corresponding multiplicative digital root of that integer. You will continue to do so until the user quits.

Additive Persistence

Additive persistence (http://mathworld.wolfram.com/AdditivePersistence.html) is a property of the sum of the digits of an integer. The sum of the digits is found, and then the summation of digits is performed on the sum, repeating until a single integer digit is reached. The number of such cycles is that integer's additive persistence. Consider the following example:

1. The beginning integer is 1234
2. The sum of its digits is 1+2+3+4 = 10
3. The integer is now 10
4. The sum of its digits is 1 + 0 = 1
5. The integer is now 1. When the value reaches a single digit, we are done. This final integer is the additive digital root.

The number of cycles is the additive persistence. The integer 1234 has an additive persistence of 2 (first sum was 10, then the second sum was 1). The final digit reached is called the integer's additive digital root. The additive digital root of 1234 is 1.

Multiplicative Persistence

The multiplicative persistence (http://mathworld.wolfram.com/MultiplicativePersistence.html) and resulting multiplicative root are determined the same way, only multiplying the digits of an integer instead of adding. For example

1. The beginning integer is 1234
2. The product of 1*2*3*4 = 24
3. The integer is now 24
4. The product of 2*4 = 8
5. The integer is now 8. When the value reaches a single digit, we are done. This final integer is the multiplicative digital root.

As before, the number of cycles is the multiplicative persistence. For 1234, the multiplicative persistence is 2, and its multiplicative digital root is 8.

Program Specification

In a file named **persistence.py** you should write a program that runs as follows:

1. Ask the user for an integer.
2. If the integer is less than 0, that is a signal to quit the program.
3. If the given integer is a single digit, report it's additive persistence and multiplicative persistence as 0 and both its additive and multiplicative root as itself.
4. If the given integer is more than a single digit, find the additive/multiplicative persistence and additive/multiplicative root of the given integer and report the results to the user
5. If the value was non-negative continue by prompting the user until they quit.

Getting Started

1. Start by understanding the process. Go through the examples in this program specification with paper and pencil, and also try it with new numbers.
2. Use what you learned about the process to write a design document. You might want to write your document in bullited form that almost resembes code (and not a big paragraph). To start, break the problem into parts. Some parts (but not all) that will be needed are:
    - gather input from the user
    - check for negative numbers (the quit condition)
    - a part that sums the digits of a number
    - a part that looks at the sum, and if it still has multiple digits, repeats the process that sums the digits of a number.
3. Create the persistence.py as usual.
4. Use the design document for code comments and to write small pieces of your code at a time.
5. Test each piece before moving on to the next.
6. Get the whole thing to work with one or the other (additive, multiplicative) before you address the other.
7. How do you get the digits of an integer? Look at a combination of integer division (//) and modulo (%) operators on integers. Try it out first. You ***must not*** convert your integers to strings to extract the digits.
8. FOR DEVELOPMENT PURPOSES ONLY, I would add some "diagnostic output" so you can be sure things are working as they should. For example, for each pass through the loop of the additive (or multiplicative) persistence, print each new integer created. ***This should be removed before you make your final submission.***

Example during development

```
What number should I use for my calculations? 999
Addition
New Value: 27
New Value: 9

Multiplication
New Value: 729
New Value: 126
New Value: 12
New Value: 2

For the integer: 999
    Additive Persistence=2, Additive Root=9
    Multiplicative Persistence=4, Multiplicative Root=2


What number should I use for my calculations? 1234
Addition
New Value: 10
New Value: 1

Multiplication
New Value: 24
New Value: 8

For the integer: 1234
    Additive Persistence=2, Additive Root=1
    Multiplicative Persistence=2, Multiplicative Root=8


What number should I use for my calculations? 2
Addition

Multiplication

For the integer: 2
    Additive Persistence=0, Additive Root=2
    Multiplicative Persistence=0, Multiplicative Root=2


What number should I use for my calculations? -1
Thanks for playing along!
```

Note that the cycles demonstrate the use of “diagnostic output” (see (8) under “Getting Started”). Your final program should not print out such diagnostic output.

Example of final deliverable

```
What number should I use for my calculations? 99199

For the integer: 99199
    Additive Persistence=3, Additive Root=1
    Multiplicative Persistence=3, Multiplicative Root=0


What number should I use for my calculations? 679

For the integer: 679
    Additive Persistence=2, Additive Root=4
    Multiplicative Persistence=5, Multiplicative Root=6


What number should I use for my calculations? 8

For the integer: 8
    Additive Persistence=0, Additive Root=8
    Multiplicative Persistence=0, Multiplicative Root=8


What number should I use for my calculations? 0

For the integer: 0
    Additive Persistence=0, Additive Root=0
    Multiplicative Persistence=0, Multiplicative Root=0


What number should I use for my calculations? -1
Thanks for playing along!
```

Notes and requirements:

Make sure that you save your programs in the correctly named files

1. Use meaningful variable names with the proper style (use_snake_casing)
2. Use meaningful constants and names where appropriate and use proper style (ALL_CAPS_AND_UNDERSCORES)
3. Every file containing python code that you submit should contain a header commen block containing three pieces of information as shown below:

```
"""
File: filename
Author: your-name
Description: one-line description of the file
"""
```

Submit your work

Submit file **persistence.py** to eLearning.