

# Python Project Number Persistence

## Assignment Overview

The aim of this project is practice the use of while loops and conditionals statements. You are going to write a program that prompts the user for an integer and then determines the additive persistence and corresponding additive root, and the multiplicative persistence, the corresponding multiplicative digital root of that integer. You will continue to do so until the user quits

## Background

There are many properties of numbers that one can investigate. The ancient Greeks were fascinated with the properties of integers, even ascribing them mystical properties.

One such property is an integer's **additive persistence** and its resulting **additive root** (<http://mathworld.wolfram.com/AdditivePersistence.html> ). Additive persistence 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. Sum 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 1. When the value reaches a single digit, we are finished. This final integer is the additive 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.

## 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 finished. This final integer is the multiplicative root.

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

### **Program Specifications**

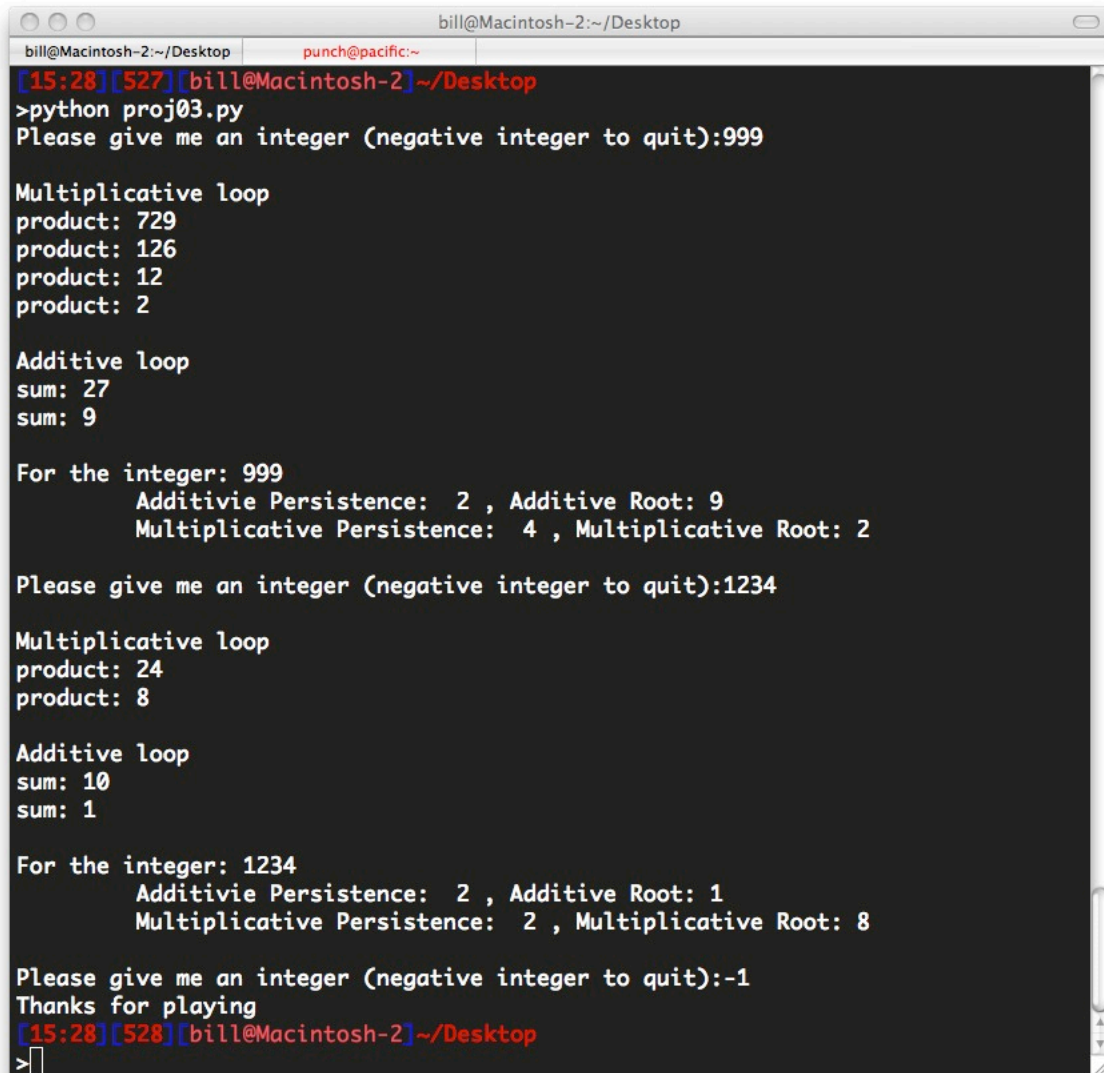
The program should run as follows.

- 1) Ask the user for an integer.
- 2) 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.
- 3) If the integer is less than 0, that is a signal to quit the program.
- 4) Otherwise, find the additive/multiplicative persistence and additive/multiplicative root of the given integer and report the results to the user
- 5) Continue by prompting the user until they quit.

### **Getting Started**

- 1) Break the problem down into parts. Some obvious parts:
  - a. gather input from the user, and check for negative numbers (the quit condition)
  - b. write a loop around the queries that can do the queries until the stop condition is reached
  - c. write a loop that can sum the digits of an integer until it reaches a single digit.
  - d. write a loop that can take the produce of the digits of an integer until it reaches a single digit, using the above approach
  - e. Get the whole thing to work with one or the other (additive, multiplicative) before you address the other
- 2) How do you get the digits of an integer? Look at a combination of division (/) and remainder(%) operators on integers. Try it out in idle first
- 3) I would add some "diagnostic output" so you can be sure things are working as they should. For each pass through the loop of the additive (or multiplicative) persistence, print each new integer created

## Example Output



```
bill@Macintosh-2:~/Desktop
[15:28][S27][bill@Macintosh-2]~/Desktop
>python proj03.py
Please give me an integer (negative integer to quit):999

Multiplicative loop
product: 729
product: 126
product: 12
product: 2

Additive loop
sum: 27
sum: 9

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

Please give me an integer (negative integer to quit):1234

Multiplicative loop
product: 24
product: 8

Additive loop
sum: 10
sum: 1

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

Please give me an integer (negative integer to quit):-1
Thanks for playing
[15:28][S28][bill@Macintosh-2]~/Desktop
>
```

Note that the first two cycles demonstrate the use of “diagnostic output” (see (4) under “Getting Started”). Your final program should not print out such diagnostic output.