**PROGRAM1:**

A digital root is the recursive sum of all the digits in a number. Given n, take the sum of the digits of n. If that value has more than one digit, continue reducing in this way until a single-digit number is produced. This is only applicable to the natural numbers.

num = input()

def droot(num):

if len(num) == 1:

return num

else:

sum = 0

for i in num:

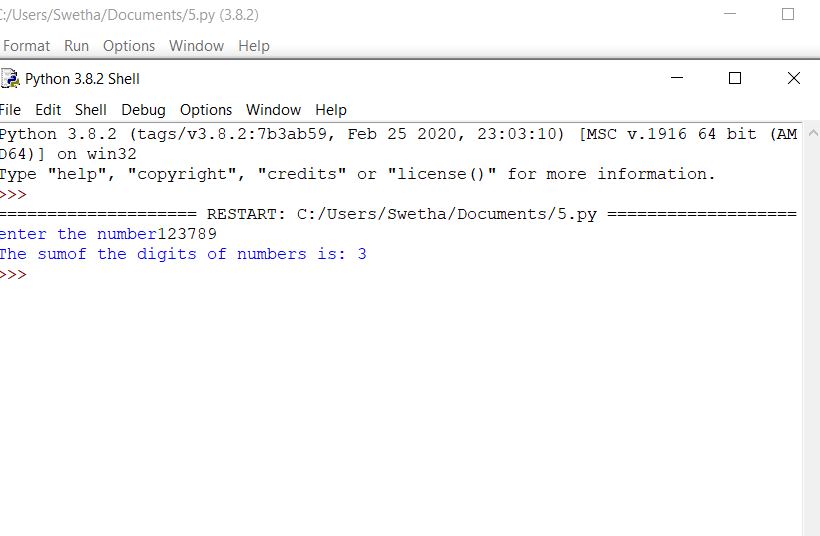
sum += int(i)

num = str(sum)

return droot(num)

print(droot(num))

**OUTPUT:**



**PROGRAM2:**

The rules for reducing the array are:  
#The first and last element say X and Y are chosen and removed from the array arr[].  
#The values X and Y are added. Z = X + Y.  
#Insert the value of Z % K into the array arr[] at the position ((N/2) + 1)th position, where N denotes the current length of the array.  
Examples:

Input: N = 5, arr[] = {1, 2, 3, 4, 5}, K = 7  
Output: 1

#include <iostream>

using namespace std;

int find\_value(int a[], int n, int k)

{

int sum = 0;

for (int i = 0; i < n; i++) {

sum += a[i];

}

return sum % k;

}

int main()

{

int n = 5, k = 3;

int a[] = { 12, 4, 13, 0, 5 };

cout << find\_value(a, n, k);

return 0;

}

**OUTPUT:**

