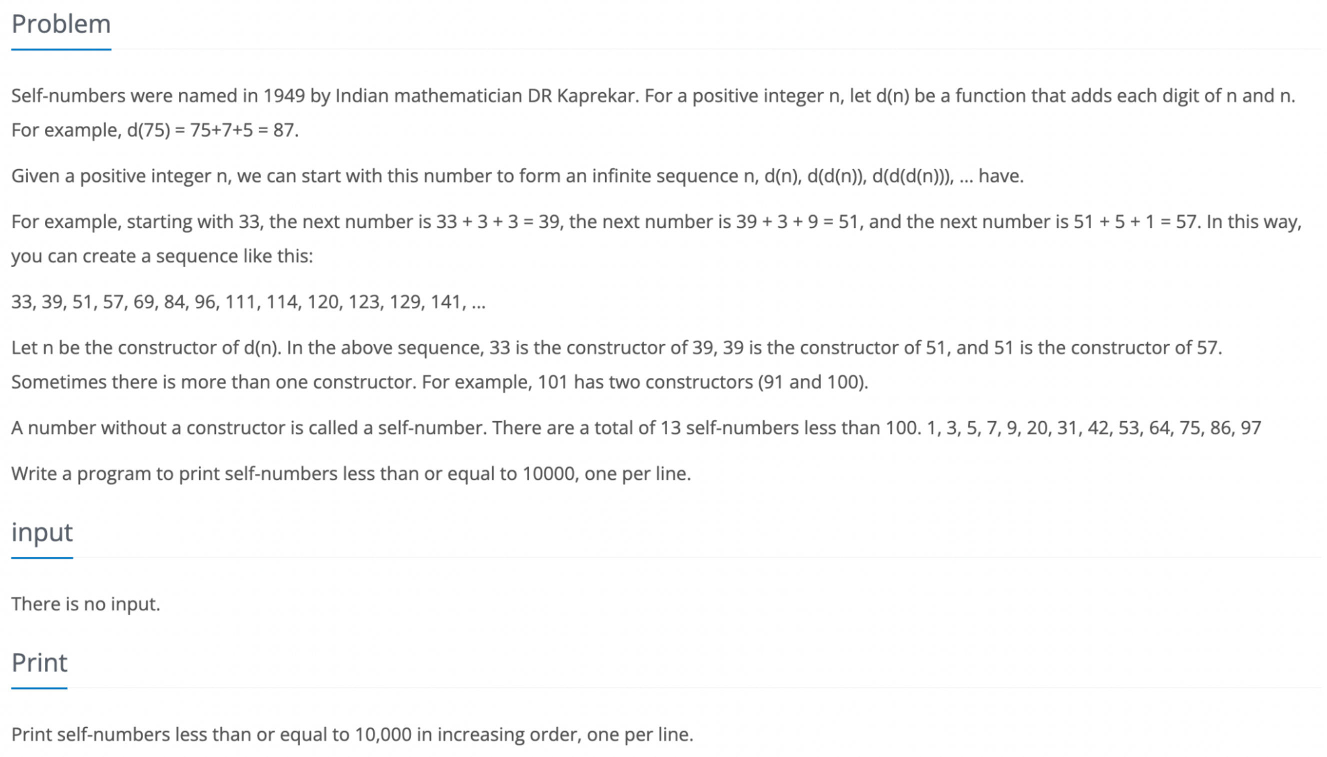
#4673



Problem

Self-numbers were named in 1949 by Indian mathematician DR Kaprekar. For a positive integer n, let d(n) be a function that adds each digit of n and n. For example, d(75) = 75+7+5 = 87.

Given a positive integer n, we can start with this number to form an infinite sequence n, d(n), d(d(n)), d(d(d(n))), ... have.

For example, starting with 33, the next number is 33 + 3 + 3 = 39, the next number is 39 + 3 + 9 = 51, and the next number is 51 + 5 + 1 = 57. In this way, you can create a sequence like this:

33, 39, 51, 57, 69, 84, 96, 111, 114, 120, 123, 129, 141, ...

Let n be the constructor of d(n). In the above sequence, 33 is the constructor of 39, 39 is the constructor of 51, and 51 is the constructor of 57. Sometimes there is more than one constructor. For example, 101 has two constructors (91 and 100).

A number without a constructor is called a self-number. There are a total of 13 self-numbers less than 100. 1, 3, 5, 7, 9, 20, 31, 42, 53, 64, 75, 86, 97

Write a program to print self-numbers less than or equal to 10000, one per line.

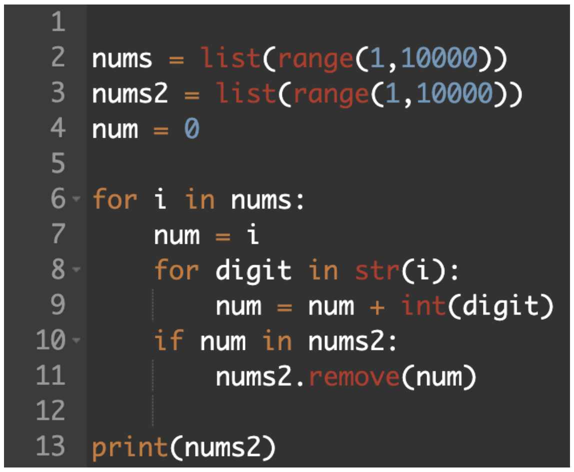
input

There is no input.

Print

Print self-numbers less than or equal to 10,000 in increasing order, one per line.

Solution:



```python

nums = list(range(1,10000))
nums2 = list(range(1,10000))
num = 0

for i in nums:
    num = i
    for digit in str(i):
        num = num + int(digit)
    if num in nums2:
        nums2.remove(num)

print(nums2)
```

The purpose of this code was to produce a list of the self-numbers within a given range. The most challenging and difficult aspect to this problem was deciding as to how to approach the problem. Should I deconstruct every integer and print the ones that could not be deconstructed? Should I first produce the self-numbers and then remove them from the list? I chose to go with the latter, as it seemed to be more efficient.