Roman_int_to_str

Statement

Roman numerals are represented by seven different symbols: `I`, `V`, `X`, `L`, `C`, `D` and `M`.

```
SymbolValue
I             1
V             5
X             10
L             50
C             100
D             500
M             1000
```

For example, `2` is written as `II` in Roman numeral, just two one's added together. `12` is written as `XII`, which is simply `X + II`. The number `27` is written as `XXVII`, which is `XX + V + II`.

Roman numerals are usually written largest to smallest from left to right. However, the numeral for four is not `IIII`. Instead, the number four is written as `IV`. Because the one is before the five we subtract it making four. The same principle applies to the number nine, which is written as `IX`. There are six instances where subtraction is used:

- `I` can be placed before `V` (5) and `X` (10) to make 4 and 9.
- `X` can be placed before `L` (50) and `C` (100) to make 40 and 90.
- `C` can be placed before `D` (500) and `M` (1000) to make 400 and 900.

Given an integer, convert it to a roman numeral.

Example 1:

```
Input: num = 3
Output: "III"
Explanation: 3 is represented as 3 ones.
```

Example 2:

```
Input: num = 58
Output: "LVIII"
Explanation: L = 50, V = 5, III = 3.
```

Example 3:

```
Input: num = 1994
Output: "MCMXCIV"
Explanation: M = 1000, CM = 900, XC = 90 and IV = 4.
```

Constraints:

- `1 <= num <= 3999`

My code

```
class Solution:
    def intToRoman(self, num: int) -> str:
        string = ""
        dic = {
            1000:{
                1:"M"
            },
            100:{
                1:"C",
                4:"CD",
                5:"D",
                9:"CM"
            },
            10:{
                1:"X",
                4:"XL",
                5:"L",
                9:"XC"
            },
            1:{
                1:"I",
                4:"IV",
                5:"V",
                9:"IX"
            }
        }
        decimal = 1
        while (decimal * 10 <= num):
            decimal *= 10
        while (decimal >= 1 and num > 0):
            quotient = int(num // decimal)
            letter = ""
            digit = 0
            #print(quotient)
#            print((dic[decimal])[1])
            if (quotient in dic[decimal]):
                letter = dic[decimal][quotient]
            else:
                for i in dic[decimal].keys():
                    if i > quotient:
                        break
                    digit = i
                #print("%i and %i and %s" %(decimal, digit, string))
                if digit > 0:
                    letter += dic[decimal][digit]
                    digit = quotient - digit
                    while (digit > 0):
                        letter += dic[decimal][1]
                        digit -= 1
            string += letter
            num = num % decimal
            decimal /= 10
        return (string)
```