Roman to Integer

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

```
Symbol Value
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 ones 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 a roman numeral, convert it to an integer.

first things first lets define the lovely dictionary given to us by the question of the symbols and the values.

```
# define the roman numerals
        translations = {
            "I": 1,
            "V": 5,
            "X": 10,
            "L": 50,
            "C": 100,
            "D": 500,
            "M": 1000
        }
```

this allows us to start replacing them with there corresponding number by initializing a variable this case `number = 0` we can replace the strings with there corresponding numbers using the .replace method in python

```
number = 0
        s = s.replace("IV", "IIII").replace("IX", "VIIII")
        s = s.replace("XL", "XXXX").replace("XC", "LXXXX")
        s = s.replace("CD", "CCCC").replace("CM", "DCCCC")
```

Basically the way this works we are replacing any letters that the roman numerals generate that would not be part of the roman numeral dictionary for example “IIII” is not 4 “IV” is then simply looping through adding the number to the translations by the chars and returning the number

```
for char in s:
            number += translations[char]
        return number
```

feel free to play around with the x, y and z test cases

```
if __name__ == '__main__':
    s = Solution()

    x = "III"
    y = "LVIII"
    z = "MCMXCIV"

    print(s.romanToInt(x))
    print(s.romanToInt(y))
    print(s.romanToInt(z))
```