Exercise for Python

1. *Palindrome Number*

Given an integer `x`, return `true` *if* `x` *is a* ***palindrome****, and* `false` *otherwise.*

Example 1:

Input: x = 121
Output: true
Explanation: 121 reads as 121 from left to right and from right to left.

Example 2:

Input: x = -121
Output: false
Explanation: From left to right, it reads -121. From right to left, it becomes 121-. Therefore it is not a palindrome.

Example 3:

Input: x = 10
Output: false
Explanation: Reads 01 from right to left. Therefore it is not a palindrome.

Constraints:

- $-2^{31} <= x <= 2^{31} - 1$

2. 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.

Example 1:

Input: s = "III"
Output: 3
Explanation: III = 3.

Example 2:

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

Example 3:

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

Constraints:

- 1 <= s.length <= 15
- s contains only the characters ('I', 'V', 'X', 'L', 'C', 'D', 'M').
- It is **guaranteed** that s is a valid roman numeral in the range [1, 3999].

3. Happy Number

Write an algorithm to determine if a number `n` is happy.

A **happy number** is a number defined by the following process:

- Starting with any positive integer, replace the number by the sum of the squares of its digits.
- Repeat the process until the number equals 1 (where it will stay), or it **loops endlessly in a cycle** which does not include 1.
- Those numbers for which this process **ends in 1** are happy.

Return `true` *if* `n` *is a happy number, and* `false` *if not*

Example 1:

Input: `n = 19`
Output: `true`
Explanation:
$1^2 + 9^2 = 82$
$8^2 + 2^2 = 68$
$6^2 + 8^2 = 100$
$1^2 + 0^2 + 0^2 = 1$

Example 2:

Input: `n = 2`
Output: `false`

Constraints:

- $1 <= n <= 2^{31} - 1$

4. Reverse String

Write a function that reverses a string. The input string is given as an array of characters ‘`s`’.

You must do this by modifying the input array in-place with `O(1)` extra memory.

Example 1:

Input: `s = ["h","e","l","l","o"]`
Output: `["o","l","l","e","h"]`

Example 2:

Input: `s = ["H","a","n","n","a","h"]`
Output: `["h","a","n","n","a","H"]`

5. *Missing Number*

Given an array `nums` containing `n` distinct numbers in the range `[0, n]`, return *the only number in the range that is missing from the array.*

Example 1:

Input: `nums = [3,0,1]`
Output: `2`
Explanation: `n = 3 since there are 3 numbers, so all numbers are in the range [0,3]. 2 is the missing number in the range since it does not appear in nums.`

Example 2:

Input: `nums = [9,6,4,2,3,5,7,0,1]`
Output: `8`
Explanation: `n = 9 since there are 9 numbers, so all numbers are in the range [0,9]. 8 is the missing number in the range since it does not appear in nums.`

Constraints:

- `n == nums.length`
- `1 <= n <= 10^4`
- `0 <= nums[i] <= n`
- All the numbers of `nums` are **unique**.

6. Bank Account

Write a Python class BankAccount with attributes like account_number, balance, date_of_opening and customer_name, and methods like deposit, withdraw, and check_balance.