Valid Anagram

Description: Given two strings s and t, return true if t is an anagram of s, and false otherwise

Qs:
- Verify the definition of an anagram (a word that contains all the same characters as the original word)
- Do s and t have to be dictionary words?
- Can I assume that s is not empty?
- What about single character strings?
- If the two strings are identical what to do?

Test Cases:
- s = "a", t = "a", returns true
- s = "bean", t = "bane", returns true
- s = "three", t = "threw", returns false
- s = "evil", t = "vile", returns true

Patterns:
- All valid anagrams are the same length
- If the strings are identical they are valid anagrams

Algorithm: (Naive Approach)
1. If the strings are identical, return true
2. If the strings aren't the same length, return false
3. Map the characters and their respective indices of s to a hashmap
4. Loop through each character in t
   i. If it is in the s hashmap, continue and remove that value from the hashmap
   ii. If it is not in the s hashmap, return false

Valid Anagram (revised)

Notes:

- My original algorithm had a few syntax errors
  - ↳ namely, when searching dictionaries by key (or value) you use square brackets
  - ↳ additionally, the del method is how you remove key-value pairs, not a .remove() (this is a constant-time operation)
- Dictionaries must have unique values
- The correct approach is to map each letter of s with its frequency
  - ↳ Ex, when s = "anagram", t = "nagaram":
    {a:3, n:1, g:1, r:1, m:1}
  - ↳ Do the same approach of subtracting from the key-value pair, return false if any of the occurrences <0

Correct Algorithm:

1. If the strings are identical, return True
2. If the strings have different lengths, return False
3. Pass each character of s to a hashmap that contains the character as the key and the # of occurences as a value
   - ↳ Handle this with if-else to avoid overwriting key-value pairs
4. Check if each character of t is in the s hashmap
   - i. If so, decrement the value
   - ii. If not, return False
5. Check if value is ever <0
   - i. If so, return False
6. Return True

Valid Anagram (solution)

```
def validAnagram(self, s, t):

    if s == t:
        return True
    if len(s) != len(t):
        return False
    s_map = {}

    for c in s:
        if c in s_map:
            s_map[c] += 1
        else:
            s_map = 1

    for c in t:
        if c in s_map:
            s_map[c] -= 1
            if s_map[c] < 0:
                return False
        else:
            return False

    return True
```