Problem -

Given two strings s and t return true if the two strings are anagrams of each other, otherwise return false

Anagram - String that contains the exact same characters as another string but the order of the characters can be different.

Working

Inputs = s = 'racecar' , t = 'carrace'
Output = True

Input s = 'jar' , t = 'jam'
Output = False

Input is one string.
Anagram is concerned w/ the individual letters.
∴ need to convert the input string into multiple strings of letters and store in a list ?

i.e. 'racecar' → ['r', 'a', 'c', 'e', 'c', 'a', 'r'] = list_1
'carrace' → ['c', 'a', 'r', 'r', 'a', 'c', 'e'] = list_2

Then iterate through list_1
and check if i is in list_2 aswell
return True

else return False

Need to check list length first
if list_s != list_t
then its definitely not an anagram
∴ can return false straightaway

✓ correct

When it comes to the for loops

```
for i in list_s :
    for j in list_t:
        if i == j:
            return True
return False
```

This is wrong. Logic & Flow ✗

list_s = race_car
List_t = car race

for i in list_s
selects r as the
character.
Then the indented loop
checks every character
in list_t against
the first character in list_s
r is the 3rd letter in list_t
∴ the loop exists and
returns True

Attempt 2 -

list_s = 'r', 'a', 'c', 'e', 'c', 'a', 'r'
list_t = 'c', 'a', 'r', 'r', 'a', 'c', 'e'

for each character in list_s
    remove that character from list_t
    If I can't return False

else return True

Python -

```
for i in list_s:
    if i in list_t:
        list_t.remove(i)
    else:
        return False  # will exist if i can't be removed from list_t
return True  # exist after the entire list has been iterated through
```

✓ Correct