
Problem Statement:

Check whether two strings *string_one* and *string_two* are anagrams (contain the same characters in any order).

Solution

```python
def valid_anagram(string_one: str, string_two: str) -> bool:
    if len(string_one) != len(string_two): return False

    counter = [0] * 26

    for char in string_one.lower():
        counter[ord(char) - ord('a')] += 1
    for char in string_two.lower():
        counter[ord(char) - ord('a')] -= 1

    for val in counter:
        if val != 0: return False
    return True
```

Step-by-Step Breakdown

1. **Input:**
    - string_one = "listen"
    - string_two = "silent"
2. **Character Count:**
    - Ensure both strings have the same length; if not, return False.
    - Initialize a list, *counter,* of size 26 (for each letter) containing zeros.
    - For each character in string_one, increment the count at ord(char) - ord('a').
        - ord(letter): returns the Unicode code representation of a character
        - Example: ord('a') will return 97.
        - Subtracting ord(a) from a lowercase letter gives the alphabetical index for that letter:
            - ord('a') - ord('a') = 97 - 97 = 0 → index 0 for 'a'
            - ord('b') - ord('a') = 98 - 97 = 1 → index 1 for 'b'
            - ord('z') - ord('a') = 122 - 97 = 25 → index 25 for 'z'

- Result after first processing step

a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	x	y	z
0	0	0	0	0	0	0	0	1	0	0	1	0	0	0	0	0	0	1	1	0	0	0	0	0	0

- For each character in string_two, decrement the count

a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	x	y	z
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

- Check if all counts in counter are zero

3. **Output:** True because no count within the final array is not equal to zero

---

Aspect	Solution One
Efficiency	O(n)
Memory Usage	Constant (O(1))
Ease of Implementation	Moderate
Scalability	Good
