📓 License Key Formatting – Full Documentation

1. Problem Statement

You are given a license key as a string s that consists of only alphanumeric characters and dashes (-). The string is split into n + 1 groups by n dashes.

You are also given an integer k. Your goal is to reformat the string such that:

- Each group contains exactly k characters,
- Except for the first group, which may be shorter but must have at least one character,
- All letters should be converted to uppercase,
- A single dash should separate groups.

2. Intuition

To satisfy the conditions:

- Remove all non-essential dashes.
- Convert all characters to uppercase.
- Then, group the characters starting from the end so that only the first group is allowed to be shorter than k.

3. Key Observations

- Dashes in the input may be in arbitrary positions and are not reliable for grouping.
- Grouping from the end ensures all groups (except possibly the first) are of exact size k.
- Working with cleaned strings simplifies the logic and reduces complexity.

4. **Approach**

Step-by-step: Clean the input:

a. Remove all dashes using replace('-', '').
b. Convert to uppercase using upper().

ii. Group the characters:

a. Start collecting groups of k characters from the end of the cleaned string.
b. Store these groups in a list.

iii. Reverse and join:

a. Since the groups were collected in reverse order, reverse the list.
b. Use '-'.join() to join the groups with dashes.

5. **Edge Cases**

- Input with no dashes (e.g., "abcd").
- All characters are dashes (e.g. "----") → Should return an empty string.
- String length less than k.
- String is already correctly formatted.
- k = 1 (every character is in its group).

6. **Complexity Analysis**

 Time Complexity:

- O(n) where n is the length of string s.
    - Removing dashes, converting to uppercase, slicing, and joining are all linear operations.

 Space Complexity:

- O(n) for storing the cleaned string and the resulting formatted string.

7. Alternative Approaches

Using string slicing and range():

Another method involves reversing the string and using a loop with range(0, len(s), k) to collect substrings of size k, then reversing back the final result.

Trade-off: Slightly more readable, but same complexity.

8. Test Cases

Test Case	Input	k	Output	Explanation
1	"5F3Z-2e-9-w"	4	"5F3Z-2E9W"	Two groups of 4 characters
2	"2-5g-3-J"	2	"2-5G-3J"	First group shorter
3	"---"	3	""	No characters left
4	"a"	1	"A"	Single character
5	"abc-def-ghi"	3	"ABC-DEF-GHI"	Already properly grouped

9. Final Thoughts

This problem tests your ability to:

- Handle string transformations,
- Maintain grouping conditions under constraints,
- Efficiently manipulate and format strings.

It's a great example of clean preprocessing, and how the order of operations (e.g., grouping from the end) simplifies logic. Always try to preprocess data into a format that makes the rest of the problem easy.