Python-Strings Advanced Questions

1. Decode Nested Encoded Strings

Implement a decoder that can handle nested encoding rules of the form `k[encoded_string]`, where `k` is a number and `encoded_string` is repeated `k` times.

📦 Input: `"3[a2[c]]"` → Output: `"accaccacc"`

2. Reconstruct String from Overlapping Substrings

Given a list of overlapping substrings, reconstruct the original string.

🧩 Input: `["abc", "bcd", "cde"]` → Output: `"abcde"`

💡 Hint: Use overlap matching logic

3. Pattern-Based String Validation

Check whether a string follows a specific character pattern, like `abba`.

🔁 Input: `pattern = "abba"`, string = `"dog cat cat dog"` → Output: `True`

🔁 Input: `pattern = "abba"`, string = `"dog cat cat fish"` → Output: `False`

4. Find Missing Ranges in Alphanumeric Sequence

Given a string of sorted alphanumeric characters, find missing ranges.

🧠 Input: `"12346789BCDF"` → Output: `["5", "A", "E"]`

📘 Hint: Use ASCII and `ord()` comparisons.

5. Find Longest Valid Mathematical Expression

From a string, extract and return the longest valid mathematical expression.

🧮 Input: `"abc1+2*3xyz4-5/2ab"` → Output: `"1+2*3"`

6. Minimize String by Removing K Characters to Get Lexicographically Smallest

Remove `k` characters to get the smallest possible string in dictionary order.

✂️ Input: `"cbacdcbc"`, `k=2` → Output: `"acdbc"`

7. Convert Sentence into CamelCase/PascalCase/Kebab-Case

Take a sentence and return it in multiple coding styles.

🎯 Input: `"convert this sentence"` → Output:

- `camelCase`: `"convertThisSentence"`
- `PascalCase`: `"ConvertThisSentence"`

- `kebab-case`: `"convert-this-sentence"`

8. Minimum Window Substring with All Characters

Find the smallest substring that contains all characters of a given target string.

🧪 Input: `s = "ADOBECODEBANC", t = "ABC"` → Output: `"BANC"`

🧠 Sliding window + frequency counter

9. Custom Alphanumeric Sorting

Sort a string by placing all letters before digits, keeping original relative order.

🔢 Input: `"a1b2c3d4"` → Output: `"abcd1234"`

📌 Hint: `isalpha()`, `isdigit()`, and stable sorting

10. Detect and Remove Cycles in Character Mapping

Given a string where each character maps to another (e.g., via dict), detect cycles and remove the minimum characters to break all cycles.

🔁 Input: `mapping = {"a": "b", "b": "c", "c": "a", "d": "e"}` → Output: `1` (remove one character from cycle `a→b→c→a`)