Palindrome Partitioning

Question: Given a string s, partition s such that every substring of the partition is a palindrome. Return all possible palindrome partitioning of s.

For example: given s = "aab"

Return [["aa","b"], ["a","a","b"]].

Solutions:

```
class Solution:
    # @param s, a string
    # @return a boolean
    def _isPalindrome(self, s):
        begin, end = 0, len(s)-1
        while begin < end:
            if s[begin] != s[end]:
                return False
            else:
                begin += 1
                end -= 1

        return True

    # @param s, a string
    # @return a list of lists of string
    def partition(self, s):
        if len(s) == 0:     return []
```

```
        if len(s) == 1:     return [[s]]

        result = []
        if self._isPalindrome(s):   result.append([s])

        for i in range(1, len(s)):
            head = s[:i]
            if not self._isPalindrome(head):
                continue

            tailPartition = self.partition(s[i:])

            result.extend([[head] + item for item in tailPartition])

        return result

Solution().partition("aab")
```