

Problem 5: You are given a string *s*, partition it in such a way that every substring is a palindrome. Return all such palindromic partitions of *s*.

```
def is_palindrome(string):
    return string == string[::-1]

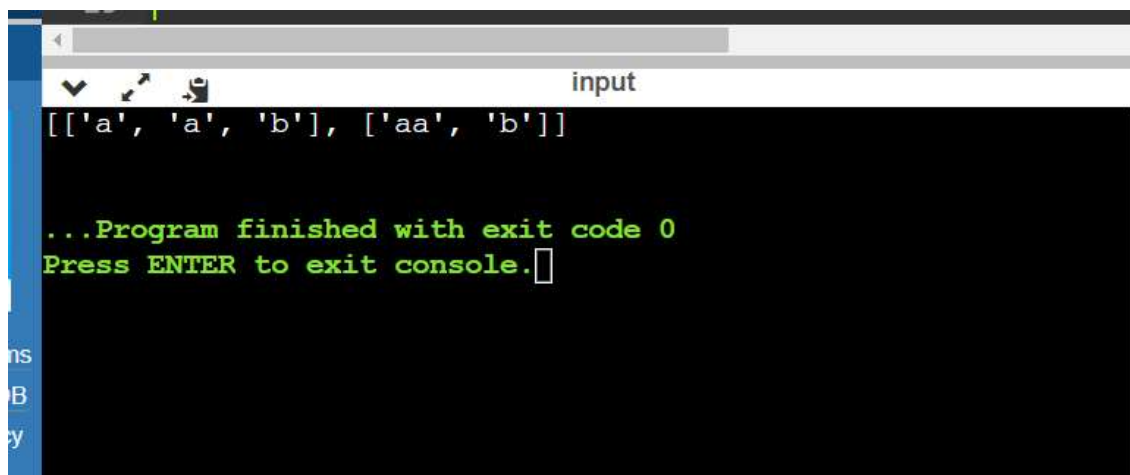
def partition_palindrome(s):
    result = []
    current_partition = []

    def backtrack(start):
        if start >= len(s):
            result.append(current_partition[:])
            return

        for end in range(start, len(s)):
            substring = s[start:end+1]
            if is_palindrome(substring):
                current_partition.append(substring)
                backtrack(end+1)
                current_partition.pop()

    backtrack(0)
    return result

s = "aab"
result = partition_palindrome(s)
print(result)
```



```
input
[['a', 'a', 'b'], ['aa', 'b']]

...Program finished with exit code 0
Press ENTER to exit console.
```