Rekursive Erzeugung aller Permutationen einer Liste.

```
def perm_lex(symbols):
    """Gib die Permutation der Elemente von symbols in lexiko-
    graphischer Ordnung aus: Der lexikographischen Ordnung wird
    _implizit_ die Ordnung der Elemente in symbols zugrunde-
    gelegt. Rückgabewert ist eine Liste aller Permutationen
    von symbols, also eine Liste von Listen; z.B.
        perm_lex([1,2]) -> [[1,2],[2,1]]
    """
    # Wenn die Liste leer ist oder nur ein
    # Element enthält, dann gibt es nur _eine_
    # Permutation (0! = 1! = 1):
    if len(symbols) <= 1:
        return [symbols]

    # Wir brauchen hier kein "else", da wir die
    # Funktion ja mit "return" verlassen, wenn die
    # Bedingung erfüllt ist: "Implizit" befinden
    # wir uns hier also in der "else-clause":


    # Rekursive Erzeugung, falls len(symbols) > 1: Der
    # Rückgabewert ist eine _Liste_ aller Permutationen,
    # wir bereiten diesen Rückgabewert einmal als
    # leere Liste vor, die wir im folgenden "anfüllen"
    # werden:
    permutations = []

    # Schleife über alle Elemente x der Liste symbols:
    for x in symbols:
        # Wir stellen eine Kopie von symbols OHNE x her ...
        without_x = deepcopy(symbols)
        without_x.remove(x)

        # ... und erzeugen die Liste aller Permutationen
        # dieser Kopie durch einen _rekursiven Aufruf_
        # der Funktion perm_lex, fügen für alle
        # so erzeugten Permutationen am Anfang das
        # Element x an, und erweitern den Rückgabewert
        # permutations um die so konstruierte Liste aller
        # Permutationen von symbols, die mit x beginnen:
        permutations += [
            [x] + pi for pi in perm_lex(without_x)
        ]

    # Rückgabewert permutations enthält nun alle Permutationen
    # von symbols in lexikographischer Ordnung: Zuerst kommen die,
    # die mit dem ersten Element von symbols beginnen, dann die,
    # die mit dem zweiten Element von symbols beginnen, usw.
    return permutations
```

Die Funktion `perm_lex` erzeugt durch eine *Rekursion* eine *lexikographisch geordnete* Liste, die alle Permutationen der gegebenen Liste `symbols` enthält. (Das Argument muß hier wirklich eine Liste sein und kein String, denn Strings haben keine .remove()-Funktion).