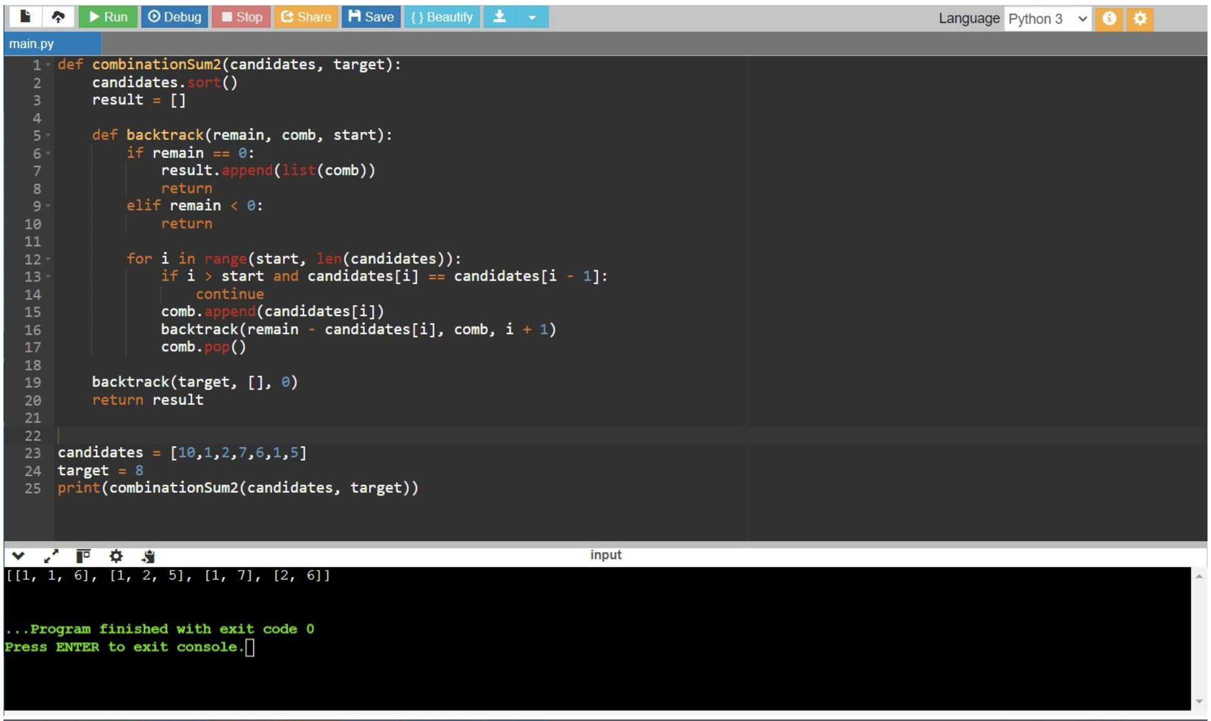
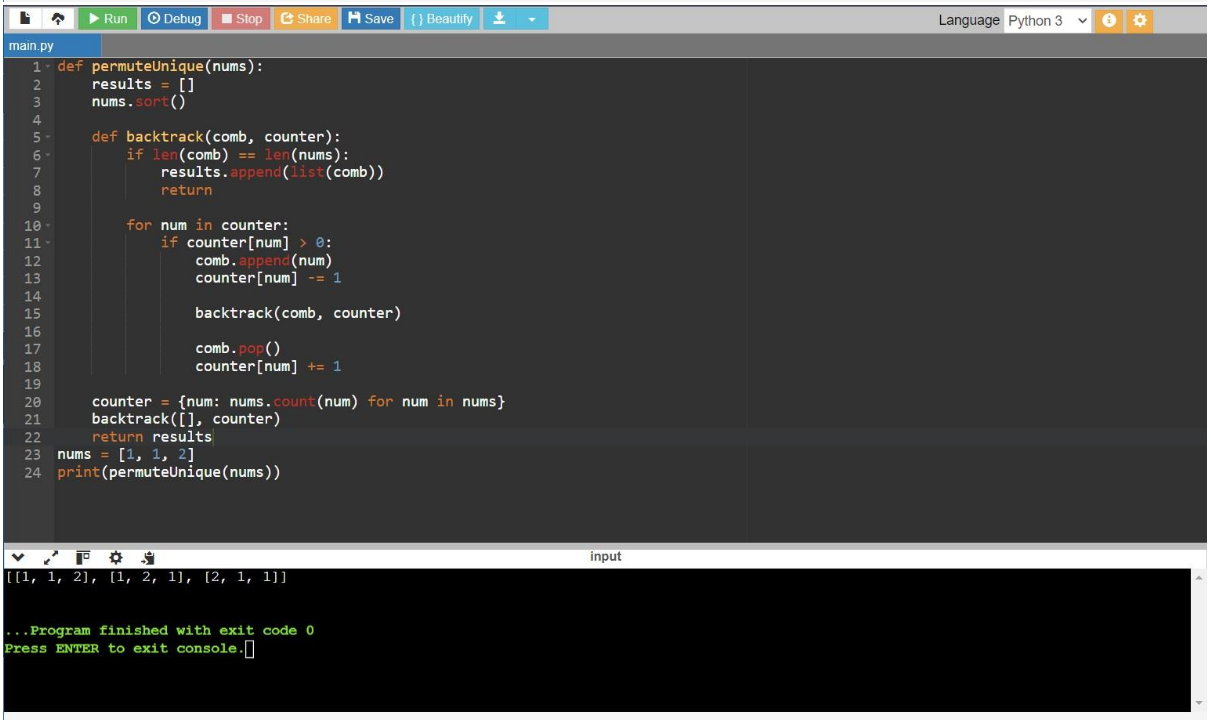


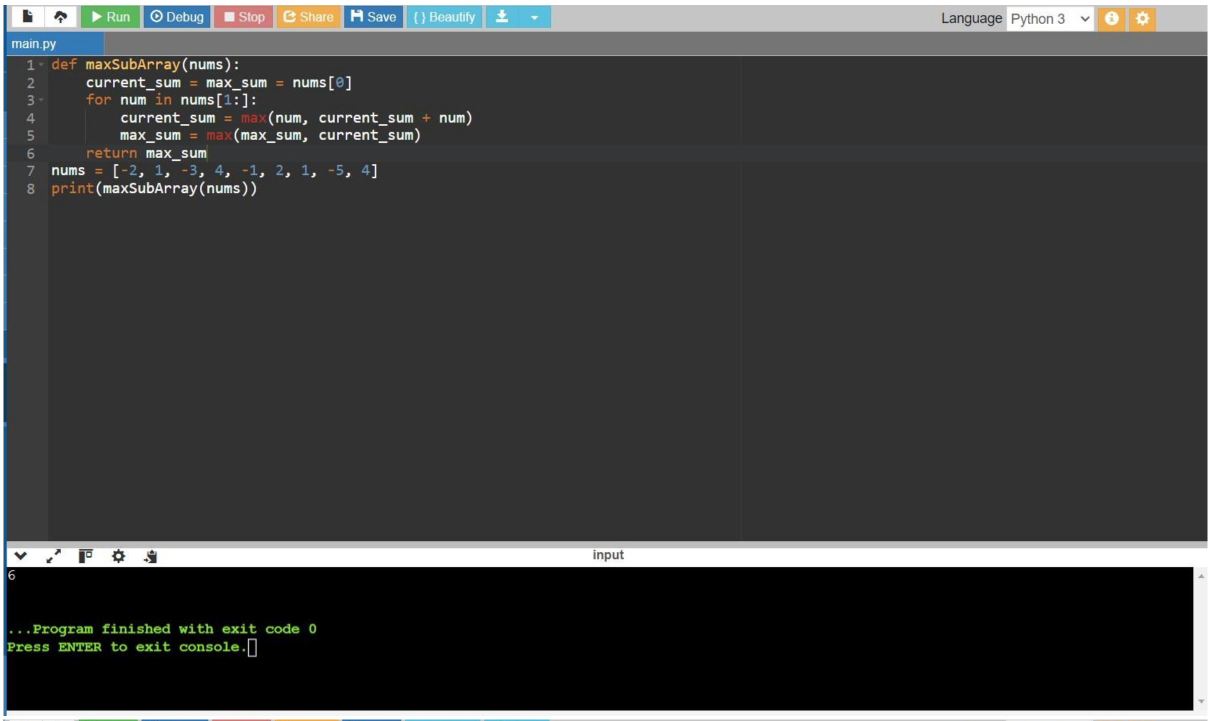
```python
def combinationSum2(candidates, target):
    candidates.sort()
    result = []

    def backtrack(remain, comb, start):
        if remain == 0:
            result.append(list(comb))
            return
        elif remain < 0:
            return

        for i in range(start, len(candidates)):
            if i > start and candidates[i] == candidates[i - 1]:
                continue
            comb.append(candidates[i])
            backtrack(remain - candidates[i], comb, i + 1)
            comb.pop()

    backtrack(target, [], 0)
    return result


candidates = [10,1,2,7,6,1,5]
target = 8
print(combinationSum2(candidates, target))
```

```
[[1, 1, 6], [1, 2, 5], [1, 7], [2, 6]]


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



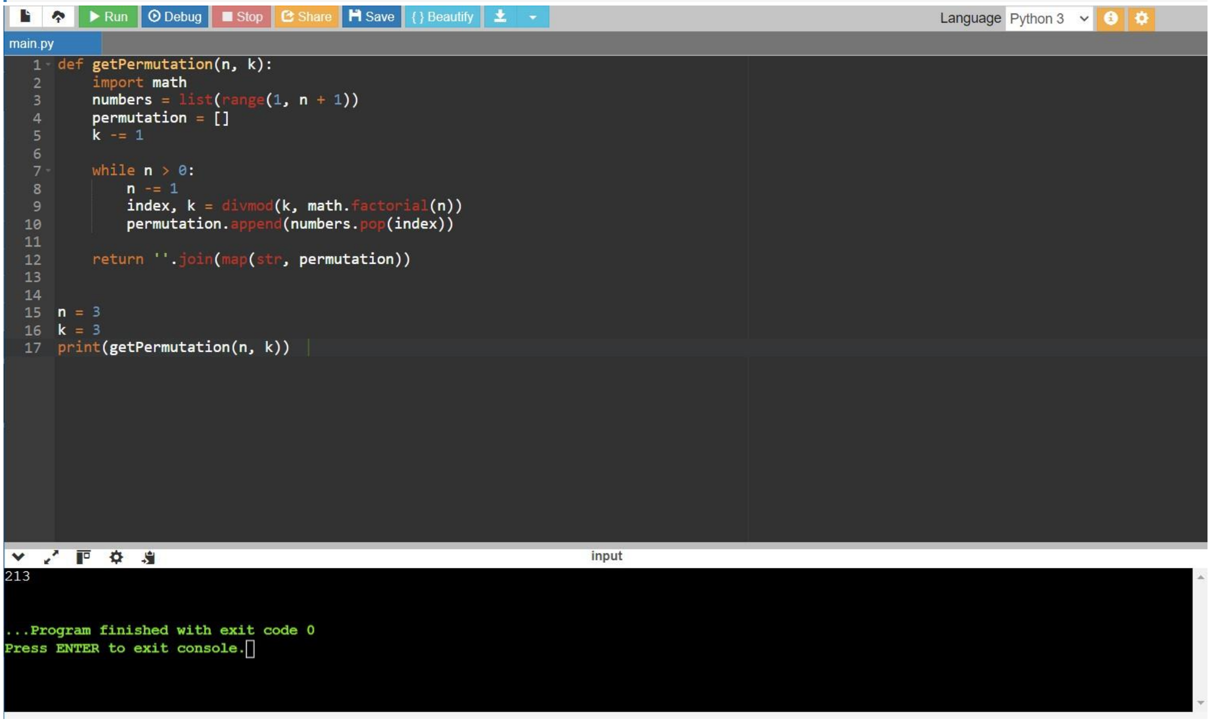
```python
def permuteUnique(nums):
    results = []
    nums.sort()

    def backtrack(comb, counter):
        if len(comb) == len(nums):
            results.append(list(comb))
            return

        for num in counter:
            if counter[num] > 0:
                comb.append(num)
                counter[num] -= 1

                backtrack(comb, counter)

                comb.pop()
                counter[num] += 1

    counter = {num: nums.count(num) for num in nums}
    backtrack([], counter)
    return results
nums = [1, 1, 2]
print(permuteUnique(nums))
```

```
[[1, 1, 2], [1, 2, 1], [2, 1, 1]]


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



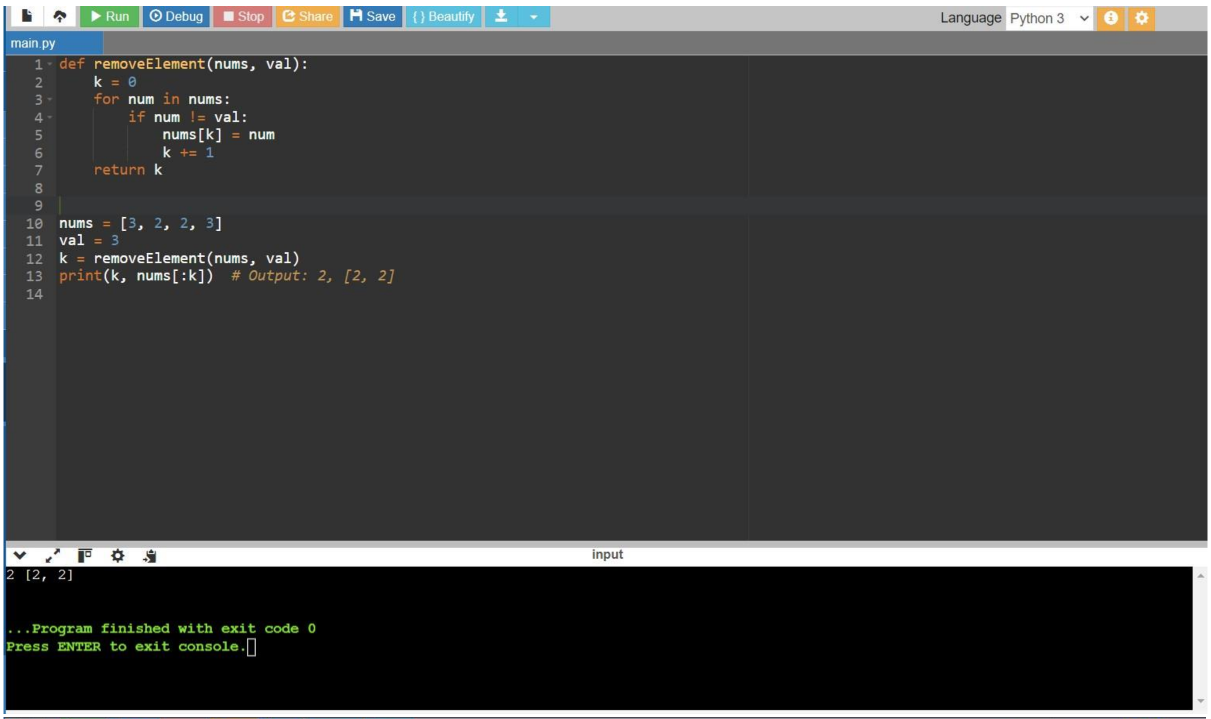
```python
def maxSubArray(nums):
    current_sum = max_sum = nums[0]
    for num in nums[1:]:
        current_sum = max(num, current_sum + num)
        max_sum = max(max_sum, current_sum)
    return max_sum
nums = [-2, 1, -3, 4, -1, 2, 1, -5, 4]
print(maxSubArray(nums))
```

```
6


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



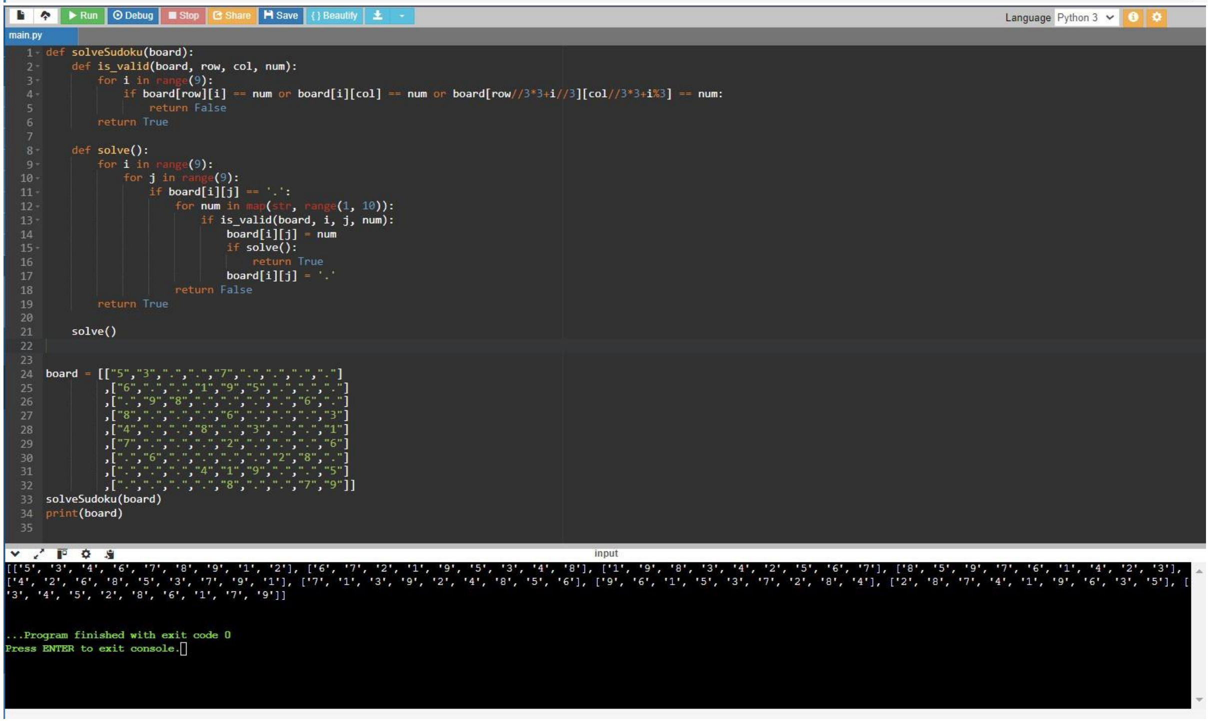
```python
def getPermutation(n, k):
    import math
    numbers = list(range(1, n + 1))
    permutation = []
    k -= 1

    while n > 0:
        n -= 1
        index, k = divmod(k, math.factorial(n))
        permutation.append(numbers.pop(index))

    return ''.join(map(str, permutation))


n = 3
k = 3
print(getPermutation(n, k))
```

```
213


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



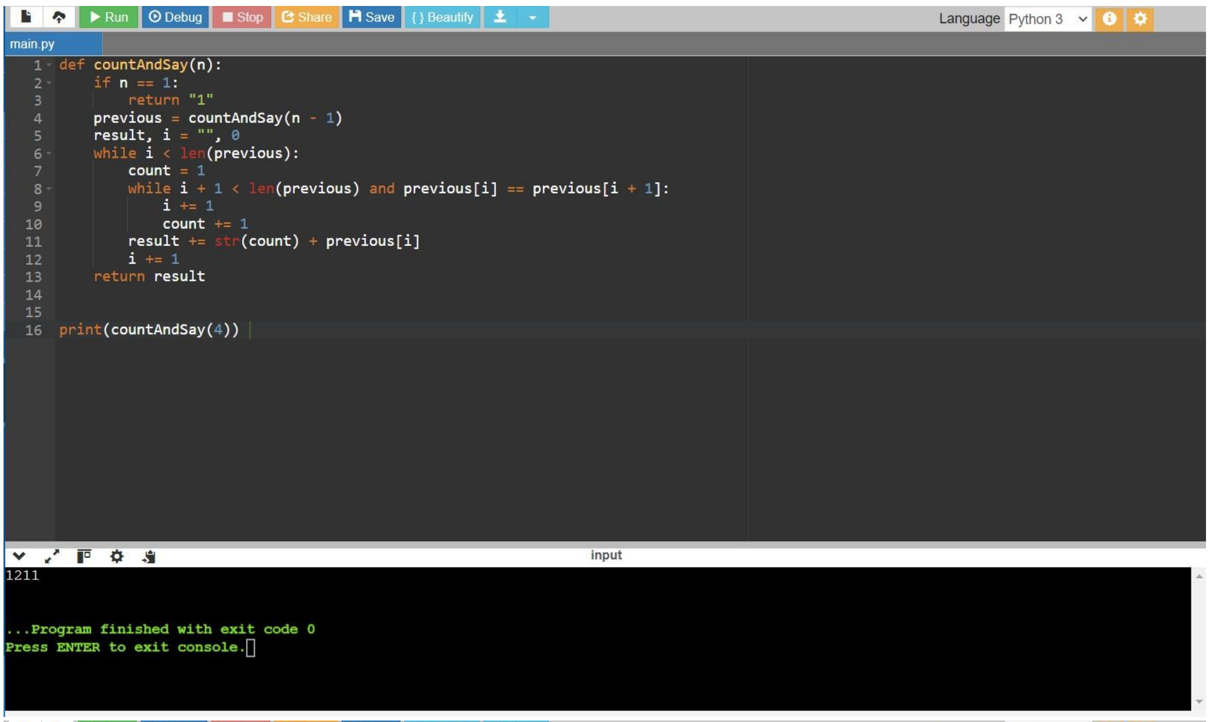
```python
def removeElement(nums, val):
    k = 0
    for num in nums:
        if num != val:
            nums[k] = num
            k += 1
    return k


nums = [3, 2, 2, 3]
val = 3
k = removeElement(nums, val)
print(k, nums[:k])  # Output: 2, [2, 2]
```

```
2 [2, 2]


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



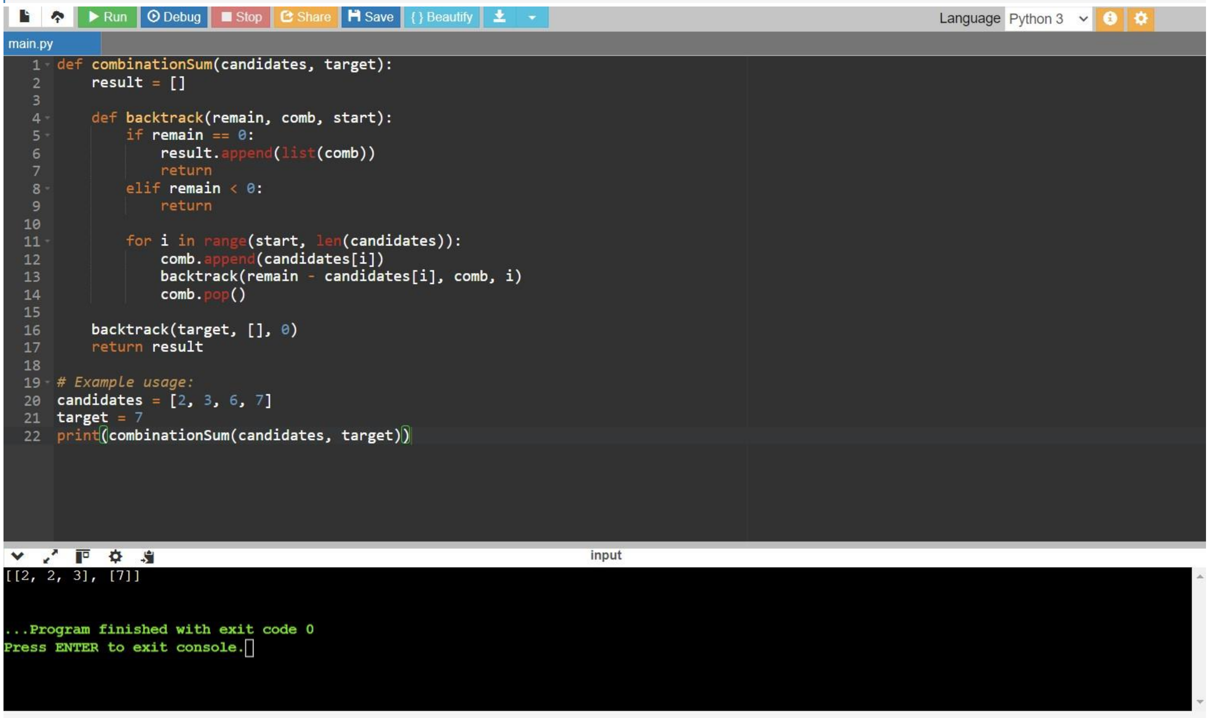
```python
def solveSudoku(board):
    def is_valid(board, row, col, num):
        for i in range(9):
            if board[row][i] == num or board[i][col] == num or board[row//3*3+i//3][col//3*3+i%3] == num:
                return False
        return True

    def solve():
        for i in range(9):
            for j in range(9):
                if board[i][j] == '.':
                    for num in map(str, range(1, 10)):
                        if is_valid(board, i, j, num):
                            board[i][j] = num
                            if solve():
                                return True
                            board[i][j] = '.'
                    return False
        return True

    solve()


board = [["5","3",".",".","7",".",".",".","."]
        ,["6",".",".","1","9","5",".",".","."]
        ,[".","9","8",".",".",".",".","6","."]
        ,["8",".",".",".","6",".",".",".","3"]
        ,["4",".",".","8",".","3",".",".","1"]
        ,["7",".",".",".","2",".",".",".","6"]
        ,[".","6",".",".",".",".","2","8","."]
        ,[".",".",".","4","1","9",".",".","5"]
        ,[".",".",".",".","8",".",".","7","9"]]
solveSudoku(board)
print(board)
```

```
[['5', '3', '4', '6', '7', '8', '9', '1', '2'], ['6', '7', '2', '1', '9', '5', '3', '4', '8'], ['1', '9', '8', '3', '4', '2', '5', '6', '7'], ['8', '5', '9', '7', '6', '1', '4', '2', '3'],
['4', '2', '6', '8', '5', '3', '7', '9', '1'], ['7', '1', '3', '9', '2', '4', '8', '5', '6'], ['9', '6', '1', '5', '3', '7', '2', '8', '4'], ['2', '8', '7', '4', '1', '9', '6', '3', '5'], [
'3', '4', '5', '2', '8', '6', '1', '7', '9']]


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



```python
def countAndSay(n):
    if n == 1:
        return "1"
    previous = countAndSay(n - 1)
    result, i = "", 0
    while i < len(previous):
        count = 1
        while i + 1 < len(previous) and previous[i] == previous[i + 1]:
            i += 1
            count += 1
        result += str(count) + previous[i]
        i += 1
    return result


print(countAndSay(4))
```

```
1211


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



```python
def combinationSum(candidates, target):
    result = []

    def backtrack(remain, comb, start):
        if remain == 0:
            result.append(list(comb))
            return
        elif remain < 0:
            return

        for i in range(start, len(candidates)):
            comb.append(candidates[i])
            backtrack(remain - candidates[i], comb, i)
            comb.pop()

    backtrack(target, [], 0)
    return result

# Example usage:
candidates = [2, 3, 6, 7]
target = 7
print(combinationSum(candidates, target))
```

```
[[2, 2, 3], [7]]


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