def find\_subset(nums, target):

    # Sort the list to help with efficient subset sum calculations

    nums.sort()

    possible\_ans = []  # List to store the current subset being checked

    helper(nums, target, 0, 0, possible\_ans)  # Call helper to start recursion

def helper(nums, target, index, current\_sum, possible\_ans):

    # Base case: If the current subset sum equals the target, print the subset

    if current\_sum == target:

        print(possible\_ans)

        return

    prev\_element = -1  # Variable to track the last element used in this loop

    # Loop through the elements starting from 'index' to explore possible subsets

    for i in range(index, len(nums)):

        # Avoid duplicates by skipping repeated elements

        if prev\_element != nums[i]:

            # If adding the current element exceeds the target, break to save computation

            if nums[i] + current\_sum > target:

                break

            # Choose the current element

            possible\_ans.append(nums[i])

            prev\_element = nums[i]  # Update prev\_element to the current element

            # Recursive call to explore further elements in the subset

            helper(nums, target, i + 1, current\_sum + nums[i], possible\_ans)

            # Backtrack by removing the last element added, exploring other subsets

            possible\_ans.pop()

if \_\_name\_\_ == "\_\_main\_\_":

    nums = [1, 2, 5, 6, 8]

    find\_subset(nums, 9)