**Approach 1: Backtracking - A recursive approach**

1)func backtrack takes 2 params : start of the combination and the combination itself

2)Base case: if length of combination is k, add it to result and return

3)Otherwise starting from ‘start’ keep finding combination upto n+1(call backtrack recursively) Once found, remove the current element and move on to find other combinations

TIme -> O(k\*n^k)

Space -> O(k)

**Code :**

def combine(self, n: int, k: int) -> List[List[int]]:

res = []

def backtrack(start, comb):

# base case

if len(comb) == k:

res.append(comb.copy())

return

# find every combination from start

for i in range(start, n+1):

# add current element

comb.append(i)

# process all comb that start with i

backtrack(i+1, comb)

# remove current element to go for next comb/backtrack

comb.pop()

# start from 1 and initially combination is []

backtrack(1, [])

return res