from typing import List

class Solution:

def letterCombinations(self, digits: str) -> List[str]:

if not digits:

return []

# Mapping of digits to letters

digit\_to\_letters = {

'2': 'abc', '3': 'def', '4': 'ghi', '5': 'jkl',

'6': 'mno', '7': 'pqrs', '8': 'tuv', '9': 'wxyz'

}

result = []

def backtrack(index, current\_combination):

# Base case: If we have processed all digits, add the current combination to the result

if index == len(digits):

result.append("".join(current\_combination))

return

# Get the letters corresponding to the current digit

current\_digit = digits[index]

letters = digit\_to\_letters[current\_digit]

# Iterate through each letter and recursively call backtrack

for letter in letters:

current\_combination.append(letter)

backtrack(index + 1, current\_combination)

current\_combination.pop() # Backtrack: remove the last added letter

# Start the backtracking process from the first digit with an empty combination

backtrack(0, [])

return result