64. It does not matter what you leave beyond the returned k (hence they are underscores).

Example 2:

Input: nums = [0,1,2,2,3,0,4,2], val = 2

Output: 5, nums = [0,1,4,0,3,_,_,_]

Explanation: Your function should return k = 5, with the first five elements of nums containing 0, 0, 1, 3, and 4.

Note that the five elements can be returned in any order.

It does not matter what you leave beyond the returned k (hence they are underscores).

Constraints:

- **0 <= nums.length <= 100**
- **0 <= nums[i] <= 50**
- **0 <= val <= 100**

Determine if a 9 x 9 Sudoku board is valid. Only the filled cells need to be validated according to the following rules:

1. **Each row must contain the digits 1-9 without repetition.**
2. **Each column must contain the digits 1-9 without repetition.**
3. **Each of the nine 3 x 3 sub-boxes of the grid must contain the digits 1-9 without repetition.**

Note:

- **A Sudoku board (partially filled) could be valid but is not necessarily solvable.**
- **Only the filled cells need to be validated according to the mentioned rules.**

Example 1:

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

Input: 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"]]

Output: true

Program:

```
def isValidSudoku(board):
    def isValidBlock(block):
        block = [num for num in block if num != '.']
        return len(block) == len(set(block))

    # Check rows
    for row in board:
        if not isValidBlock(row):
            return False

    # Check columns
    for col in range(9):
        if not isValidBlock([board[row][col] for row in range(9)]):
            return False

    # Check 3x3 sub-boxes
    for boxRow in range(3):
```

```
        for boxCol in range(3):
            block = [
                board[r][c]
                for r in range(boxRow*3, boxRow*3 + 3)
                for c in range(boxCol*3, boxCol*3 + 3)
            ]
            if not isValidBlock(block):
                return False

    return True

# Example usage
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"]
]
print(isValidSudoku(board))  # Output: True
```

Output:

```
"C:\Program Files\Python312\python.exe" "C:\Work Space\DAA\DAA COADS.PYTHON\program 64.py"
True

Process finished with exit code 0
```

Time Complexity:

O(1)