

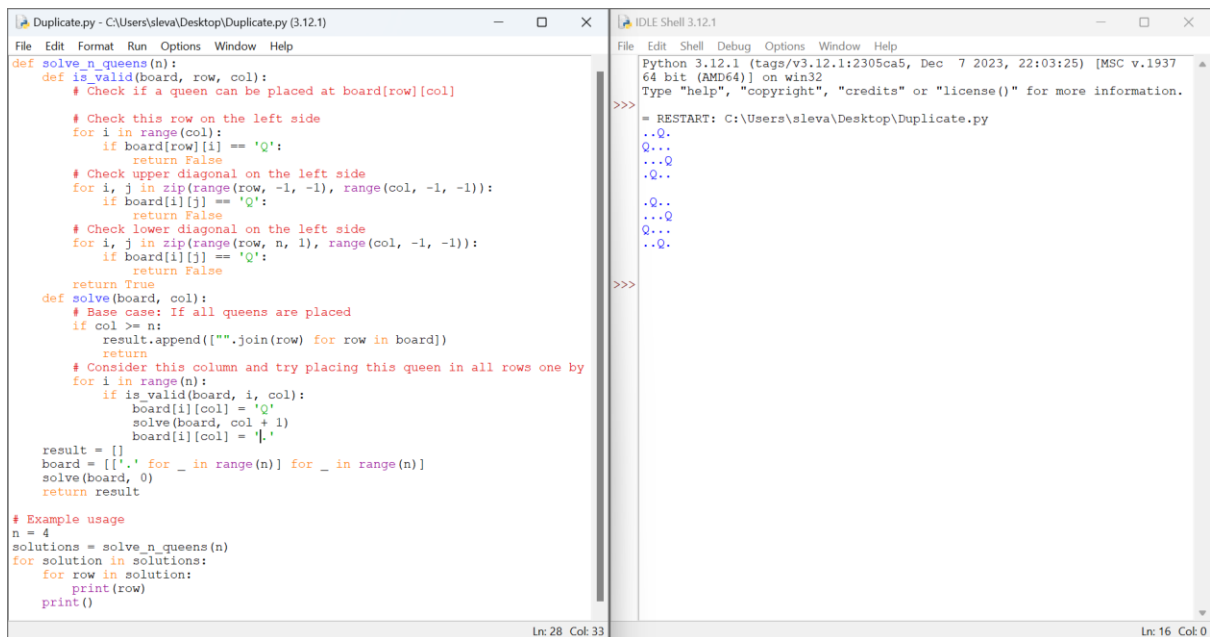
1. Sudoku Board

```
*Duplicate.py - C:\Users\sleval\Desktop\Duplicate.py (3.12.1)*
File Edit Format Run Options Window Help
def is_valid_sudoku(board):
    # Check rows
    for row in board:
        if not is_valid_unit(row):
            return False
    # Check columns
    for col in zip(*board):
        if not is_valid_unit(col):
            return False
    # Check 3x3 sub-boxes
    for i in range(0, 9, 3):
        for j in range(0, 9, 3):
            if not is_valid_box(board, i, j):
                return False
    return True
def is_valid_unit(unit):
    """Check if a row or column is valid."""
    unit = [num for num in unit if num != '.']
    return len(unit) == len(set(unit))
def is_valid_box(board, start_row, start_col):
    """Check if a 3x3 sub-box is valid."""
    box = []
    for i in range(3):
        for j in range(3):
            num = board[start_row + i][start_col + j]
            if num != '.':
                box.append(num)
    return is_valid_unit(box)
# Example Usage
sudoku_board = [
    ["5", "3", ".", ".", "7", ".", ".", ".", "."],
    ["6", ".", ".", "1", "9", "5", ".", ".", "."],
    [".", "9", "8", ".", ".", ".", "6", ".", "."],
    ["8", ".", ".", "6", ".", ".", ".", "3", "."],
    ["4", ".", ".", "8", ".", "3", ".", ".", "1"],
    ["7", ".", ".", "2", ".", ".", ".", "6", "."],
    [".", "6", ".", ".", "2", "8", ".", ".", "."],
    ["9", ".", "4", "1", "9", ".", "5", ".", "."],
    [".", ".", ".", "8", ".", ".", "7", "9", "."]
]
for i in range(0, len(sudoku_board)):
    >>>
Python 3.12.1 (tags/v3.12.1:2305ca5, Dec 7 2023, 22:03:25) [MSC v.1937
64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>> = RESTART: C:\Users\sleval\Desktop\Duplicate.py
>>> ['5', '3', '.', '.', '7', '.', '.', '.', '.']
>>> ['6', '1', '9', '5', '.', '.', '.', '.']
>>> ['.', '9', '8', '.', '.', '.', '6', '.', '.']
>>> ['8', '6', '.', '3', '.', '.', '.', '.']
>>> ['4', '8', '3', '1', '9', '5', '6', '7']
>>> ['7', '2', '8', '6', '3', '1', '5', '4']
>>> ['.', '6', '2', '8', '7', '4', '1', '9', '5']
>>> ['9', '4', '1', '9', '5', '6', '3', '8']
>>> ['.', '8', '7', '9', '6', '3', '5', '1']
>>> The sudoku board is True
Ln: 28 Col: 29
Ln: 15 Col: 0
```

2. Sudoku Solver

```
*Duplicate.py - C:\Users\sleval\Desktop\Duplicate.py (3.12.1)*
File Edit Format Run Options Window Help
def solve_sudoku(board):
    def is_valid(board, row, col, num):
        # Check if the number is not repeated in the current row
        for i in range(9):
            if board[row][i] == num:
                return False
        # Check if the number is not repeated in the current column
        for i in range(9):
            if board[i][col] == num:
                return False
        # Check if the number is not repeated in the current 3x3 sub-box
        start_row, start_col = 3 * (row // 3), 3 * (col // 3)
        for i in range(3):
            for j in range(3):
                if board[start_row + i][start_col + j] == num:
                    return False
        return True
    def solve():
        for row in range(9):
            for col in range(9):
                if board[row][col] == '.':
                    for num in map(str, range(1, 10)):
                        if is_valid(board, row, col, num):
                            board[row][col] = num
                            if solve():
                                return True
                            board[row][col] = '.'
                    return False
        return True
    solve()
# Example usage
sudoku_board = [
    ["5", "3", ".", ".", "7", ".", ".", ".", "."],
    ["6", ".", ".", "1", "9", "5", ".", ".", "."],
    [".", "9", "8", ".", ".", ".", "6", ".", "."],
    ["8", ".", ".", "6", ".", ".", ".", "3", "."],
    ["4", ".", ".", "8", ".", "3", ".", ".", "1"],
    ["7", ".", ".", "2", ".", ".", ".", "6", "."],
    [".", "6", ".", ".", "2", "8", ".", ".", "."],
    ["9", ".", "4", "1", "9", ".", "5", ".", "."],
    [".", ".", ".", "8", ".", ".", "7", "9", "."]
]
Ln: 1 Col: 0
Ln: 14 Col: 0
>>>
Python 3.12.1 (tags/v3.12.1:2305ca5, Dec 7 2023, 22:03:25) [MSC v.1937
64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>> = RESTART: C:\Users\sleval\Desktop\Duplicate.py
>>> ['5', '3', '.', '.', '7', '.', '.', '.', '.']
>>> ['6', '1', '9', '5', '3', '4', '8', '2']
>>> ['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']
```

3.N-Queens



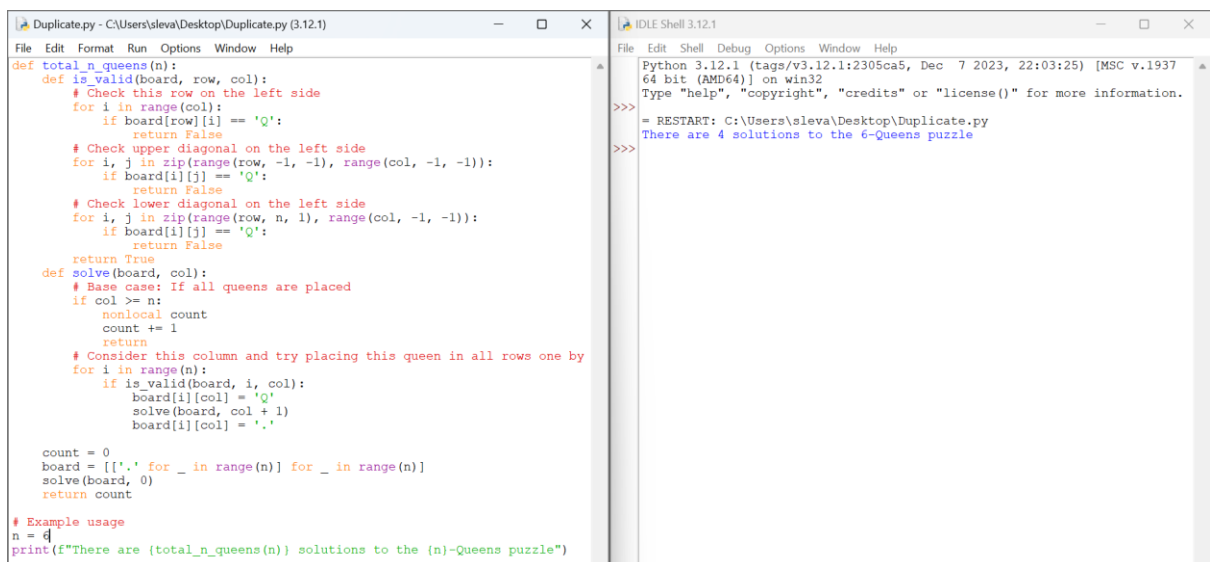
```
def solve_n_queens(n):
    def is_valid(board, row, col):
        # Check if a queen can be placed at board[row][col]

        # Check this row on the left side
        for i in range(col):
            if board[row][i] == 'Q':
                return False
        # Check upper diagonal on the left side
        for i, j in zip(range(row, -1, -1), range(col, -1, -1)):
            if board[i][j] == 'Q':
                return False
        # Check lower diagonal on the left side
        for i, j in zip(range(row, n, 1), range(col, -1, -1)):
            if board[i][j] == 'Q':
                return False
        return True
    def solve(board, col):
        # Base case: If all queens are placed
        if col >= n:
            result.append("".join(row for row in board))
            return
        # Consider this column and try placing this queen in all rows one by one
        for i in range(n):
            if is_valid(board, i, col):
                board[i][col] = 'Q'
                solve(board, col + 1)
                board[i][col] = '.'
        result = []
        board = [['.' for _ in range(n)] for _ in range(n)]
        solve(board, 0)
        return result

    # Example usage
    n = 4
    solutions = solve_n_queens(n)
    for solution in solutions:
        for row in solution:
            print(row)
        print()
```

```
>>>
Python 3.12.1 (tags/v3.12.1:2305ca5, Dec 7 2023, 22:03:25) [MSC v.1937
64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
= RESTART: C:\Users\sleval\Desktop\Duplicate.py
..Q.
Q...
...Q
..Q.
>>>
..Q.
...Q
Q...
..Q.
>>>
```

4. N-Queens-II

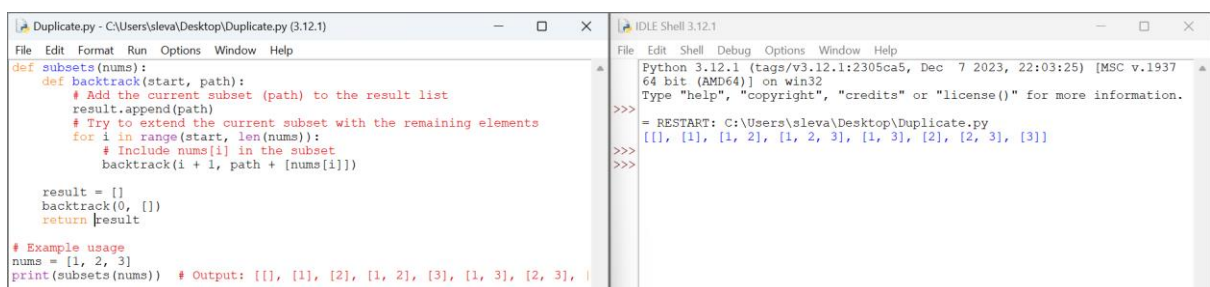


```
def total_n_queens(n):
    def is_valid(board, row, col):
        # Check this row on the left side
        for i in range(col):
            if board[row][i] == 'Q':
                return False
        # Check upper diagonal on the left side
        for i, j in zip(range(row, -1, -1), range(col, -1, -1)):
            if board[i][j] == 'Q':
                return False
        # Check lower diagonal on the left side
        for i, j in zip(range(row, n, 1), range(col, -1, -1)):
            if board[i][j] == 'Q':
                return False
        return True
    def solve(board, col):
        # Base case: If all queens are placed
        if col >= n:
            nonlocal count
            count += 1
            return
        # Consider this column and try placing this queen in all rows one by one
        for i in range(n):
            if is_valid(board, i, col):
                board[i][col] = 'Q'
                solve(board, col + 1)
                board[i][col] = '.'
        count = 0
        board = [['.' for _ in range(n)] for _ in range(n)]
        solve(board, 0)
        return count

    # Example usage
    n = 4
    print(f"There are {total_n_queens(n)} solutions to the {n}-Queens puzzle")
```

```
>>>
Python 3.12.1 (tags/v3.12.1:2305ca5, Dec 7 2023, 22:03:25) [MSC v.1937
64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
= RESTART: C:\Users\sleval\Desktop\Duplicate.py
There are 4 solutions to the 6-Queens puzzle
>>>
```

5.Subsets



```
def subsets(nums):
    def backtrack(start, path):
        # Add the current subset (path) to the result list
        result.append(path)
        # Try to extend the current subset with the remaining elements
        for i in range(start, len(nums)):
            # Include nums[i] in the subset
            backtrack(i + 1, path + [nums[i]])

    result = []
    backtrack(0, [])
    return result

# Example usage
nums = [1, 2, 3]
print(subsets(nums)) # Output: [[], [1], [2], [1, 2], [3], [1, 3], [2, 3], [1, 2, 3]]
```

```
>>>
Python 3.12.1 (tags/v3.12.1:2305ca5, Dec 7 2023, 22:03:25) [MSC v.1937
64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
= RESTART: C:\Users\sleval\Desktop\Duplicate.py
[[], [1], [1, 2], [1, 2, 3], [1, 3], [2], [2, 3], [3]]
>>>
```

6.Longest Palindrome Substring

```
Duplicate.py - C:\Users\sleva\Desktop\Duplicate.py (3.12.1)
File Edit Format Run Options Window Help
def longest_palindrome(s):
    if not s:
        return ""

    start, end = 0, 0

    def expand_around_center(left, right):
        while left >= 0 and right < len(s) and s[left] == s[right]:
            left -= 1
            right += 1
        return left + 1, right - 1

    for i in range(len(s)):
        l1, r1 = expand_around_center(i, i) # Odd length palindromes
        l2, r2 = expand_around_center(i, i + 1) # Even length palindromes

        if r1 - l1 > end - start:
            start, end = l1, r1
        if r2 - l2 > end - start:
            start, end = l2, r2

    return s[start:end + 1]

# Example usage
s = "babad"
print(longest_palindrome(s)) # Output: "bab" or "aba"

IDLE Shell 3.12.1
File Edit Shell Debug Options Window Help
Python 3.12.1 (tags/v3.12.1:2305ca5, Dec 7 2023, 22:03:25) [MSC v.1937
64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
= RESTART: C:\Users\sleva\Desktop\Duplicate.py
bab
>>>
```