lab3 Реализовать решатель Судоку.

def solve\_sudoku(board):

    """

    Основная функция решения судоку.

    Использует алгоритм обратногоPropagation, пытаясь заполнить цифры и проверяя соответствие правилам.

    Параметр:

    board -- двумерный список 9x9, представляющий поле судоку, пустые ячейки обозначены 0

    Возвращает:

    True если судоку имеет решение, False если нет

    """

    # Найти первую пустую ячейку на поле

    empty = find\_empty(board)

    if not empty:

        return True  # Если пустых ячеек нет, судоку решено

    row, col = empty

    # Попытаться заполнить цифрами от 1 до 9

    for num in range(1, 10):

        # Проверить, может ли цифра быть в этой позиции

        if is\_valid(board, row, col, num):

            # Если может, заполнить ячейку

            board[row][col] = num

            # Рекурсивно попытаться заполнить следующую пустую ячейку

            if solve\_sudoku(board):

                return True

            # Если рекурсивный вызов возвращает False, значит эта цифра не подходит, откатываем и пробуем следующую

            board[row][col] = 0

    # Если ни одна из цифр 1-9 не подходит, судоку не имеет решения

    return False

def find\_empty(board):

    """

    Найти первую пустую ячейку (значение 0) на поле

    Параметр:

    board -- поле судоку

    Возвращает:

    (row, col) координаты пустой ячейки, если пустых ячеек нет, возвращает None

    """

    for row in range(9):

        for col in range(9):

            if board[row][col] == 0:

                return (row, col)

    return None

def is\_valid(board, row, col, num):

    """

    Проверить, может ли цифра быть в указанной позиции

    Параметры:

    board -- поле судоку

    row -- номер строки (0-8)

    col -- номер столбца (0-8)

    num -- проверяемая цифра (1-9)

    Возвращает:

    True если цифра может быть в этой позиции, False иначе

    """

    # Проверка строки

    for x in range(9):

        if board[row][x] == num:

            return False

    # Проверка столбца

    for x in range(9):

        if board[x][col] == num:

            return False

    # Проверка 3x3 блока

    # Определение начала блока

    start\_row = (row // 3) \* 3

    start\_col = (col // 3) \* 3

    for i in range(3):

        for j in range(3):

            if board[start\_row + i][start\_col + j] == num:

                return False

    return True

# Пример поля судоку

example\_board = [

    [5, 3, 0, 0, 7, 0, 0, 0, 0],

    [6, 0, 0, 1, 9, 5, 0, 0, 0],

    [0, 9, 8, 0, 0, 0, 0, 6, 0],

    [8, 0, 0, 0, 6, 0, 0, 0, 3],

    [4, 0, 0, 8, 0, 3, 0, 0, 1],

    [7, 0, 0, 0, 2, 0, 0, 0, 6],

    [0, 6, 0, 0, 0, 0, 2, 8, 0],

    [0, 0, 0, 4, 1, 9, 0, 0, 5],

    [0, 0, 0, 0, 8, 0, 0, 7, 9]

]

# Решение примера судоку

if solve\_sudoku(example\_board):

    print("Судоку решено:")

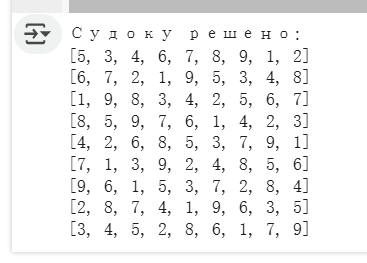
    for row in example\_board:

        print(row)

else:

    print("Судоку не имеет решения")

Результаты выполнения кода



**Инструкции по выполнению кода**

resolve\_sudoku(board) — это основная функция, которая решает задачу судоку с помощью алгоритма возврата:

Найти первое пустое место на доске

Попробовать заполнить его числом от 1 до 9

Проверить, является ли заполненное число допустимым

Если допустимо, рекурсивно вызвать себя, чтобы попытаться заполнить следующее место

Если рекурсивный вызов не удается, вернуться и попробовать следующее число

Функция find\_empty(board) используется для поиска первого пустого места на доске (позиция со значением 0).

Функция is\_valid(board, row, col, num) используется для проверки, является ли заполнение указанного числа в указанной позиции допустимым:

Проверить, существует ли уже такое же число в строке

Проверить, существует ли уже такое же число в столбце

Проверить, существует ли уже такое же число в сетке 3x3