# Sudoku Solver using Backtracking

# Function to print the board

Def print\_board(board):

For I in range(9):

If I % 3 == 0 and I != 0:

Print(“- - - - - - - - - - - - -“)

For j in range(9):

If j % 3 == 0 and j != 0:

Print(“ | “, end=””)

If j == 8:

Print(board[i][j])

Else:

Print(str(board[i][j]) + “ “, end=””)

# Function to find an empty cell

Def find\_empty(board):

For I in range(9):

For j in range(9):

If board[i][j] == 0: # 0 means empty cell

Return (I, j)

Return None

# Check if a number is valid

Def is\_valid(board, num, pos):

# Check row

For j in range(9):

If board[pos[0]][j] == num and pos[1] != j:

Return False

# Check column

For I in range(9):

If board[i][pos[1]] == num and pos[0] != i:

Return False

# Check 3x3 box

Box\_x = pos[1] // 3

Box\_y = pos[0] // 3

For I in range(box\_y \* 3, box\_y \* 3 + 3):

For j in range(box\_x \* 3, box\_x \* 3 + 3):

If board[i][j] == num and (I, j) != pos:

Return False

Return True

# Solve function using backtracking

Def solve(board):

Empty = find\_empty(board)

If not empty:

Return True # Solved

Else:

Row, col = empty

For num in range(1, 10):

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

Board[row][col] = num

If solve(board):

Return True

Board[row][col] = 0 # Reset and try next number

Return False

# Example unsolved Sudoku board (0 represents empty cells)

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]

]

Print(“Unsolved Sudoku:”)

Print\_board(board)

Solve(board)

Print(“\nSolved Sudoku:”)

Print\_board(board)