Sudoku

```python
#!/usr/bin/env python3

#-------------------------------------------------------------------------------
# Input a Sudoku puzzle, and output both the given puzzle and its solution
#-------------------------------------------------------------------------------
# Input Format : a 9 x 9 square of numbers, with zero representing blank
#-------------------------------------------------------------------------------

def Solve( grid, row = 0, col = 0 ) :

    # Attempt to solve the puzzle 'grid', starting from the cell ( row, col );
    # return a tuple  ( success, solution )  where, if a solution was found,
    # 'success' is True and 'solution' is this solution, or if no such solution
    # was found, 'success' is False and 'solution' is the unchanged grid 'grid'

    if row == 9 :
        return ( True, grid )

    else :
        ( nextrow, nextcol ) = ( row, col + 1 ) if col < 8 else ( row + 1, 0 )

        if grid[ row ][ col ] != 0 :
            return Solve( grid, nextrow, nextcol )

        else :
            for num in range( 1, 10 ) :
                if IsValid( grid, row, col, num ) :
                    ( success, solution ) = \
                        Solve( Update( grid, row, col, num ), nextrow, nextcol )
                    if success :
                        return ( True, solution )

            return ( False, Update( grid, row, col, 0 ) )

#-------------------------------------------------------------------------------

def IsValid( grid, row, col, num ) :

    # Is it valid to place 'num' in cell ( row, col ) of 'grid' ?

    #------ check if 'num' occurs in row 'row'
    for c in range( 9 ) :
        if grid[ row ][ c ] == num :
            return False
    #------ check if 'num' occurs in column 'col'
    for r in range( 9 ) :
        if grid[ r ][ col ] == num :
            return False
    #------ check if 'num' occurs in the 3 x 3 box containing ( row, col )
    boxrow = ( row // 3 ) * 3
    boxcol = ( col // 3 ) * 3
    for r in range( boxrow, boxrow + 3 ) :
        for c in range( boxcol, boxcol + 3 ) :
            if grid[ r ][ c ] == num :
                return False

    return True

#-------------------------------------------------------------------------------
```

```python
#-------------------------------------------------------------------------------

def Update( grid, row, col, num ) :

    # The grid 'grid', with 'num' now in cell ( row, col )

    grid[ row ][ col ] = num

    return grid

#-------------------------------------------------------------------------------

def ReadGrid( filename ) :  # no error checking

    # Input the puzzle from file 'filename' and return it
    # as a grid : a 9-item list of 9-item lists

    filehandle = open( filename, "r" )

    return [ [ int( n ) for n in filehandle.readline( ).split( ) ]
             for row in range( 9 ) ]

#-------------------------------------------------------------------------------

def WriteGrid( title, grid ) :

    # Output the string 'title' and the grid 'grid'

    print( "\n%s\n" % ( title ) )

    for row in range( 9 ) :
        if row % 3 == 0 :
            print( " +-------+-------+-------+" )
        for col in range( 9 ) :
            if col % 3 == 0 :
                print( " |", end = "" )
            if grid[ row ][ col ] == 0 :
                print( "  ", end = "" )
            else :
                print( " %i" % ( grid[ row ][ col ] ), end = "" )
        print( " |" )
    print( " +-------+-------+-------+" )

#-------------------------------------------------------------------------------

from sys import argv

startgrid = ReadGrid( argv[ 1 ] )  # no error checking

WriteGrid( "PROBLEM:", startgrid )

( success, solution ) = Solve( startgrid )

if success :
    WriteGrid( "SOLUTION:", solution )
else :
    print( "\nNO SOLUTION" )

#-------------------------------------------------------------------------------
```

Sudoku

```
$ cat easy
5 4 1 0 6 3 8 0 0
0 0 0 0 0 1 0 0 7
7 0 0 0 0 4 1 0 5
8 0 7 1 0 0 0 0 3
2 0 4 0 7 0 9 0 8
9 0 0 0 0 8 5 0 2
4 0 2 5 0 0 0 0 1
1 0 0 3 0 0 0 0 0
0 0 9 2 1 0 7 5 4


$ sudoku easy

PROBLEM:

 +-------+-------+-------+
 | 5 4 1 |   6 3 | 8     |
 |       |     1 |     7 |
 | 7     |     4 | 1   5 |
 +-------+-------+-------+
 | 8   7 | 1     |     3 |
 | 2   4 |   7   | 9   8 |
 | 9     |     8 | 5   2 |
 +-------+-------+-------+
 | 4   2 | 5     |     1 |
 | 1     | 3     |       |
 |     9 | 2 1   | 7 5 4 |
 +-------+-------+-------+

SOLUTION:

 +-------+-------+-------+
 | 5 4 1 | 7 6 3 | 8 2 9 |
 | 6 2 8 | 9 5 1 | 4 3 7 |
 | 7 9 3 | 8 2 4 | 1 6 5 |
 +-------+-------+-------+
 | 8 5 7 | 1 9 2 | 6 4 3 |
 | 2 3 4 | 6 7 5 | 9 1 8 |
 | 9 1 6 | 4 3 8 | 5 7 2 |
 +-------+-------+-------+
 | 4 6 2 | 5 8 7 | 3 9 1 |
 | 1 7 5 | 3 4 9 | 2 8 6 |
 | 3 8 9 | 2 1 6 | 7 5 4 |
 +-------+-------+-------+
```

```
$ cat fiendish
0 7 6 0 4 0 8 0 0
0 0 2 0 0 0 0 9 0
0 0 8 0 6 9 0 3 4
0 0 0 1 0 0 2 0 5
0 0 0 4 0 7 0 0 0
8 0 1 0 0 5 0 0 0
3 9 0 8 5 0 1 0 0
0 8 0 0 0 0 6 0 0
0 0 5 0 1 0 9 8 0


$ sudoku fiendish

PROBLEM:

 +-------+-------+-------+
 |   7 6 |   4   | 8     |
 |     2 |       |   9   |
 |     8 |   6 9 |   3 4 |
 +-------+-------+-------+
 |       | 1     | 2   5 |
 |       | 4   7 |       |
 | 8   1 |     5 |       |
 +-------+-------+-------+
 | 3 9   | 8 5   | 1     |
 |   8   |       | 6     |
 |     5 |   1   | 9 8   |
 +-------+-------+-------+

SOLUTION:

 +-------+-------+-------+
 | 9 7 6 | 5 4 3 | 8 2 1 |
 | 4 3 2 | 7 8 1 | 5 9 6 |
 | 5 1 8 | 2 6 9 | 7 3 4 |
 +-------+-------+-------+
 | 7 4 3 | 1 9 8 | 2 6 5 |
 | 6 5 9 | 4 2 7 | 3 1 8 |
 | 8 2 1 | 6 3 5 | 4 7 9 |
 +-------+-------+-------+
 | 3 9 7 | 8 5 6 | 1 4 2 |
 | 1 8 4 | 9 7 2 | 6 5 3 |
 | 2 6 5 | 3 1 4 | 9 8 7 |
 +-------+-------+-------+
```