Red Ant Media LLP

Technical Assessment

Name: Jane karuniya .J

Roll no: 1832025

M.Sc. Data Science

Question: 03

Cathy works as an editor in a magazine. She is responsible for the games and puzzles section in the magazine. But unfortunately, she doesn’t have time to go about creating a puzzle on her own every month. Help her by creating a sudoku validator.

1. It will be a standard 9 X 9 sudoku board
2. Each row must contain the digits 1-9 without repetition.
3. Each column must contain the digits 1-9 without repetition.
4. Each of the nine 3 x 3 sub-boxes of the grid must contain the digits 1-9 without repetition.
5. Provide an interface where there will be an empty board.
6. Cathy can type in the squares that she wants with numbers that she wants to.
7. When she clicks on validate, the solution should tell her whether the partially entered board by her is a valid sudoku board or not

Language Used: **Python**

* To Print like a Sudoku Board

Code:

def print\_board(board):

'''Prints the board'''

boardString = ""

for i in range(9):

for j in range(9):

boardString += str(board[i][j]) + " "

if (j+1)%3 == 0 and j != 0 and (j+1) != 9:

boardString += "| "

if j == 8:

boardString += "\n"

if j == 8 and (i+1)%3 == 0 and (i+1) != 9:

boardString += "- - - - - - - - - - - \n"

print(boardString)

* To Find an empty cell and returns its position as a tuple

Code:

def find\_empty (board):

for i in range (9):

for j in range (9):

if board[i][j] == 0:

return i,j

* Whether a number is valid in that cell, returns a bool

Code:

def valid(board, pos, num):

'''Whether a number is valid in that cell, returns a bool'''

for i in range(9):

if board[i][pos[1]] == num and (i, pos[1]) != pos: #make sure it isn't the same number we're checking for by comparing coords

return False

for j in range(9):

if board[pos[0]][j] == num and (pos[0], j) != pos: #Same row but not same number

return False

start\_i = pos[0] - pos[0] % 3 #ex. 5-5%3 = 3 and thats where the grid starts

start\_j = pos[1] - pos[1] % 3

for i in range(3):

for j in range(3): #adds i and j as needed to go from start of grid to where we need to be

if board[start\_i + i][start\_j + j] == num and (start\_i + i, start\_j + j) != pos:

return False

return True

* Solves the Sudoku board via the backtracking algorithm

Code:

def solve(board):

'''Solves the Sudoku board via the backtracking algorithm'''

empty = find\_empty(board)

if not empty: #no empty spots are left so the board is solved

return True

for nums in range(9):

if valid(board, empty,nums+1):

board[empty[0]][empty[1]] = nums+1

if solve(board): #recursive step

return True

board[empty[0]][empty[1]] = 0 #this number is wrong so we set it back to 0

return False

ARCHITECTURE OF CODE:

* Creating a Sudoku Board
* Print a Sudoku Board
* Checking for a Correct Answer

Code:

if \_\_name\_\_ == '\_\_main\_\_':

board = [

[0, 0, 0, 0, 0, 0, 0, 0, 0],

[0, 0, 0, 0, 0, 0, 0, 0, 0],

[0, 0, 0, 0, 0, 0, 0, 0, 0],

[0, 0, 0, 0, 0, 0, 0, 0, 0],

[0, 0, 0, 0, 0, 0, 0, 0, 0],

[0, 0, 0, 0, 0, 0, 0, 0, 0],

[0, 0, 0, 0, 0, 0, 0, 0, 0],

[0, 0, 0, 0, 0, 0, 0, 0, 0],

[0, 0, 0, 0, 0, 0, 0, 0, 0]

]

ch1 = 'y'

while(ch1 == 'y'):

print("Enter Your Choice ")

print("1. Creating A Sudoku Board ")

print("2. Print A Board")

print("3. Check for Answer")

ch = int(input('[1/2/3] : '))

if(ch == 1):

#Cathy Input Board

number = int(input("Enter the no of input numbers : "))

for i in range(number):

row = int(input('Enter Row No. to change :'))

column = int(input('Enter Column No. to change :'))

board[row][column] = int(input('Enter the new number : '))

print\_board(board)

elif(ch == 2):

print\_board(board)

else:

solve(board)

print\_board(board)

ch1 = input('Do you want to Continue [y/n] : ')

OUTPUT:



