**SUDOKU SOLVER**



Name: ***Ujjwal Kakar and Vartika Sood***

*CERTIFICATE*

**CLASS: XI-F YEAR: 2021-2022**

This is to certify that Investigatory Project is successfully completed by ***Ujjwal Kakar and Vartika Sood*** of ***Class: XI F*** for the academic year 2021-2022.

Internal Examiner (Subject Teacher)

Date: 15 /1 / 22 Department of: COMPUTER SCIENCE

*Acknowledgement:*

We, ***Ujjwal Kakar and Vartika Sood*** of class ***XI- F***

would like to express our sincere gratitude to our computer science teacher ***Ms. Gurmeet Kaur, PGT COMPUTER SCIENCE***, for her constant support and ideas for growth in our endeavour to pursue thoughtful and functional programs as we study python.

We also made use well of online tools to help guide us with more complex parts of syntax and documentation in labriaries, such as tkinter.

*INDEX*

|  |  |
| --- | --- |
| 1 | *Brief Overview of Project* |
| 2 | ***Limitations of Project*** |
| 3 | ***Source Code of Project*** |
| 4 | ***Output Screens*** |
| 4 | ***Future Enhancement of Project*** |
| 5 | ***Bibliography*** |

*Brief Overview of Project:*

**Definition:**

A project which aims to complete any given sudoku puzzle with the prerequisite numbers given

**Objective:**

Sudoku Solving, by making user input appropriate numbers in the grid using coordinates and then using logic functions to solve it

**Contribution / Team members:**

Vartika Sood, Ujjwal Kakar

**Team Detail:**

The project “Sudoku Solver” is developed by Vartika Sood and Ujjwal Kakar, it took approx. 5 days to develop this project, working 2 hours a day. All modules completed by us only as per our view and knowledge.

**Teamwork and Division:**

Logical Expertise: Vartika Sood

Visual Presentation: Ujjwal Kakar

User Input Interface: Ujjwal Kakar

Modular Division of program: Vartika Sood

**Reason for choosing the Topic:**

Sudoku is a classic puzzle, ranging from various difficulty levels.

Often, we come across one of those pesky ones, which we cannot solve, after even hours of hard work and effort, in such cases we have developed a program that can seamlessly solve that specific troublesome puzzle for you.

It will not only help you to find a solution to the Sudoku, but will also give you the peace of mind, that the puzzle had once taken away from you.

So, you can move on happily to your next puzzle…

**Hardware Requirements:**

* A Computer/Laptop with
* Operating System-Windows 7 or above
* 5 GB free disk space.

**Software Requirements:**

* Python 3.9
* Tkinter library
* Rich library

*Limitations of project:*

* It is not web based project
* No provision to print hard copies.

*Source Code of Project:*

**The main screen python code:**

**###########################################################**

**# SUDOKU SOLVER #**

**###########################################################**

**import sys**

**from doku import \***

**try:**

**import rich**

**except:**

**print("Rich library not installed, Please install rich and then open")**

**sys.exit()**

**from rich import print**

**from rich.console import Console**

**Console()**

**print("""[blue]##########################################################**

**#[/blue] [bold red]SUDOKU SOLVER[/ bold red] [blue]#**

**###########################################################[/blue]""",end="\n\n")**

**print("[green]Welcome to Sudoku Solver [/green]")**

**print("""[green]This program allows you to solve the classic Sudoku Puzzle.**

**The Puzzle is famous with its 9x9 grid where no number repeats in its row, column or 3x3 square. [/green]\n""")**

**print("[blue]This project has been developed in two formats, one with GUI and another using the Console, please choose which would you prefer \n1)GUI \n2)Console[/blue]")**

**while True:**

**global progType**

**progType = input("Enter value[1 or 2]: ")**

**if progType=='1' or progType=='2':**

**break**

**print("[red]Invalid Input, Try Again[/red]")**

**if progType=='1':**

**import doku\_gui**

**if progType=='2':**

**enterInput(myBoard)**

**solveBoard(myBoard)**

**printBoard(myBoard)**

**The Sudoku solving python code:**

**from rich import print**

**# myBoard = [[0, 4, 0, 7, 0, 0, 1, 3, 0],**

**# [0, 0, 2, 0, 0, 0, 6, 0, 0],**

**# [0, 0, 0, 4, 2, 0, 0, 0, 0],**

**# [6, 0, 0, 0, 0, 2, 0, 0, 3],**

**# [2, 3, 1, 0, 7, 0, 0, 8, 0],**

**# [4, 0, 0, 3, 1, 0, 0, 0, 0],**

**# [0, 7, 0, 0, 0, 8, 0, 0, 0],**

**# [0, 0, 6, 0, 3, 0, 0, 0, 4],**

**# [8, 9, 0, 0, 5, 0, 0, 0, 6]]**

**myBoard = [[' ' for \_ in range(9)] for \_\_ in range(9)]**

**def enterInput(board):**

**print(""" Enter your values in following format:**

**[blue]x y v[/blue]**

**[red]x[/red] is the x coordinate from left**

**[red]y[/red] is the y coordinate from top**

**[red]v[/red] is number in that position**

**All values are between 1 and 9**

**Type [red]'solve'[/red] to solve \n""")**

**c = 1**

**while True:**

**nStr = '1 2 3 4 5 6 7 8 9'**

**while True:**

**inp = input(" Enter Value "+str(c)+': ').split()**

**if inp[0].lower() == 'solve':**

**c = 0**

**break**

**try:**

**if inp[0] in nStr and inp[1] in nStr and inp[2] in nStr:**

**inp = [ int(inp[0]) , int(inp[1]) , int(inp[2]) ]**

**putVal( board , inp[0] , inp[1] , inp[2])**

**break**

**except: pass**

**print("Invalid Input Try Again")**

**if c==0:**

**break**

**c+=1**

**def putVal(board, x, y, v):**

**board[y-1][x-1] = v**

**for x in board:**

**print('[',end='')**

**for j in x:**

**print(j,end=" ")**

**print(']')**

**def isValid(board, row, col, num):**

**for i in range(9):**

**if board[row][i] == num:**

**return False**

**for i in range(9):**

**if board[i][col] == num:**

**return False**

**c\_row = row - row%3**

**c\_col = col - col%3**

**for i in range(c\_row, c\_row+3):**

**for j in range(c\_col, c\_col+3):**

**if board[i][j] == num:**

**return False**

**return True**

**def solveBoard(board):**

**for i in range(9):**

**for j in range(9):**

**if board[i][j] == ' ':**

**for num in range(1,10):**

**if isValid(board, i, j, num):**

**board[i][j] = num**

**result = solveBoard(board)**

**if result == True:**

**return True**

**else:**

**board[i][j] = ' '**

**return False**

**return True**

**def printBoard(board):**

**for line in board:**

**print(line)**

**input()**

**The GUI python code:**

**from tkinter import \***

**from doku import \***

**from time import sleep**

**#Root main window and main sudoku variable**

**tkinterSudoku = [[0 for \_ in range(10)] for \_\_ in range(10)]**

**root = Tk()**

**#attributes of root window**

**root.title("Sudoku Solver")**

**root.geometry('725x700')**

**root.resizable(False , False)**

**root.configure(bg="Yellow")**

**root.lift()**

**root.attributes('-topmost',True)**

**root.after\_idle(root.attributes,'-topmost',False)**

**#3 chilren of root window**

**TitleFrame = Frame(root)**

**MainFrame = Frame(root,bg="black", height=40, padx=1,pady=1)**

**BottomFrame = Frame(root,bg="Yellow",width = 100, height=40)**

**#initialise string variable**

**vival = [[StringVar(root, value=" ") for \_ in range(10)] for \_\_ in range(10)]**

**#Set the values of CornerGrids**

**for j in range(1,10):**

**vival[0][j].set(j)**

**for j in range(1,10):**

**vival[j][0].set(" ABCDEFGHI"[j])**

**#Function for editing value of Grid - Used for Button**

**def EditStringVar(a,i,j,val):**

**a[int(j)+1][int(i)].set(str(val))**

**print(i,int(j)+1)**

**#Function for converting vival to board**

**def convertBoard():**

**takenBoard = [[int(vival[i][j].get())**

**if vival[i][j].get().isnumeric()**

**else vival[i][j].get()**

**for j in range(1,10)]**

**for i in range(1,10)]**

**solveBoard(takenBoard)**

**print(takenBoard)**

**for i in range(9):**

**for j in range(9):**

**vival[i+1][j+1].set(takenBoard[i][j])**

**root.update()**

**sleep(0.013)**

**#Create the sudoku itself by making labels and assigning String Variable**

**def MakeSudoku(a):**

**for i in range(10):**

**for j in range(10):**

**tkinterSudoku[i][j] = Label(MainFrame, width=4, height=1, bg="white",textvariable=a[i][j],padx=1,pady=1,font=("Comic Sans MS",18)).grid(row=i,column=j,padx=1,pady=2)**

**def buttonCommand():**

**EditStringVar(**

**vival,**

**int( option1.get() ),**

**"ABCDEFGHI".index(option2.get()),**

**int( option3.get() ) )**

**#Makes Title and the Solve Button**

**title = Label(TitleFrame, width=24 , height=2 , text="Sudoku Solver",font="Algerian 28",bg='Light green')**

**solveButton = Button(TitleFrame, bg="orange",text="Solve", width=10 , height=2,font=("Comic Sans MS 20",20), command=convertBoard)**

**#makes middle part = Sudoku**

**MakeSudoku(vival)**

**#Initialises Base variables for Option menu**

**one2nine = list("123456789")**

**a2i = list("ABCDEFGHI")**

**option1=StringVar(root,value="1")**

**option2=StringVar(root,value="A")**

**option3=StringVar(root,value="1")**

**#Creating Option Menu**

**Opt1 = OptionMenu(BottomFrame,option1,\*one2nine)**

**Opt2 = OptionMenu(BottomFrame,option2,\*a2i )**

**Opt3 = OptionMenu(BottomFrame,option3,\*one2nine)**

**Opt1.config(width=3,height=2,bg='light blue',font=("Algerian",18))**

**Opt2.config(width=3,height=2,bg='light blue',font=("Algerian",18))**

**Opt3.config(width=3,height=2,bg='light blue',font=("Algerian",18))**

**#Creates button**

**OptButton = Button(BottomFrame, width=10,height=2,text="Put Value",**

**command=buttonCommand,font=("Algerian",18))**

**Opt1.pack(expand=True,side=LEFT,padx=40,pady=35)**

**Opt2.pack(expand=True,side=LEFT,padx=40,pady=35)**

**Opt3.pack(expand=True,side=LEFT,padx=40,pady=35)**

**OptButton.pack(side=LEFT)**

**#Makes root's grid**

**TitleFrame.grid(row=0 , column=0)**

**MainFrame.grid(row=1 , column=0,pady=20)**

**BottomFrame.grid(row=2 , column=0)**

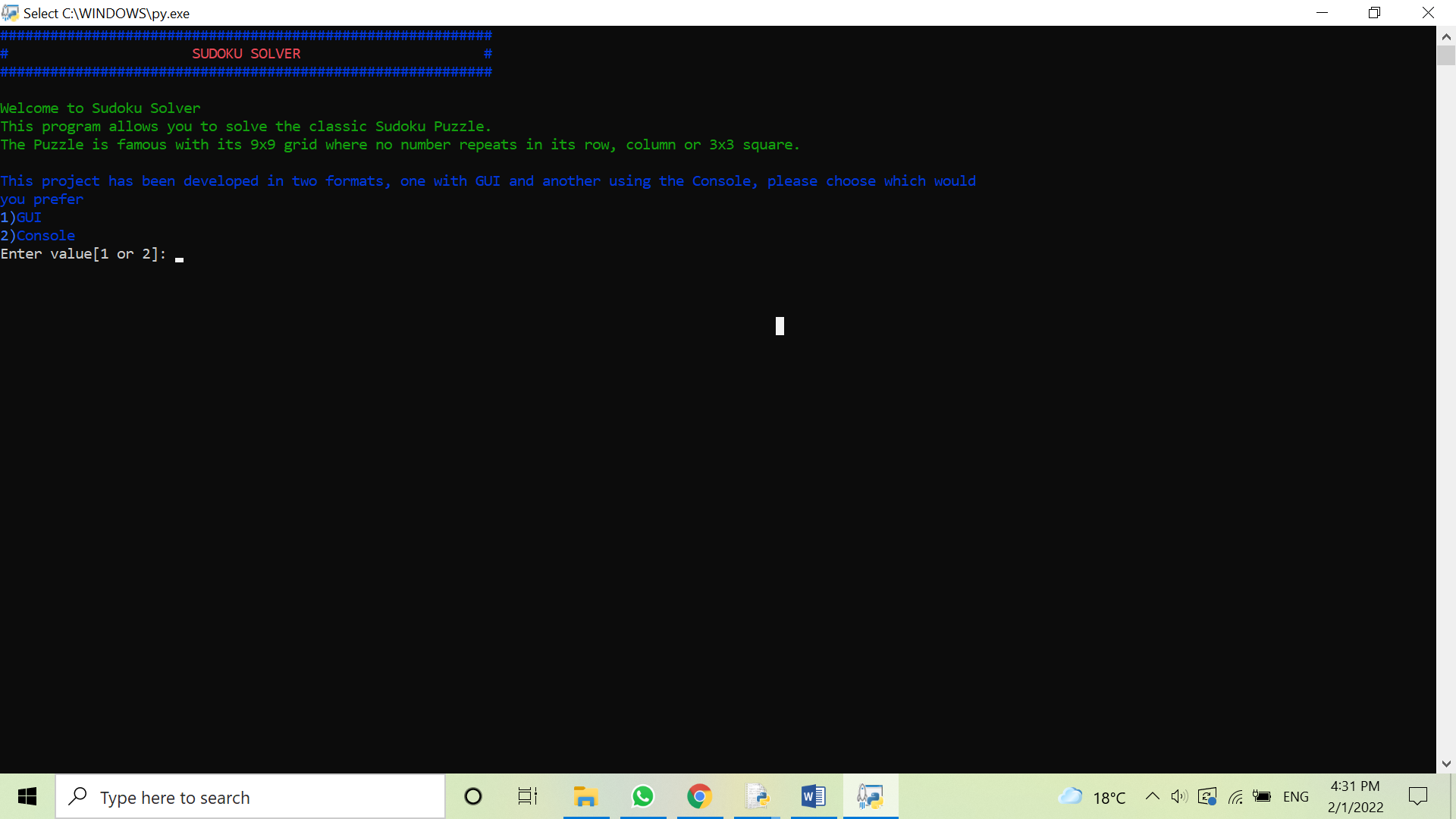
**#make titleframe's grid**

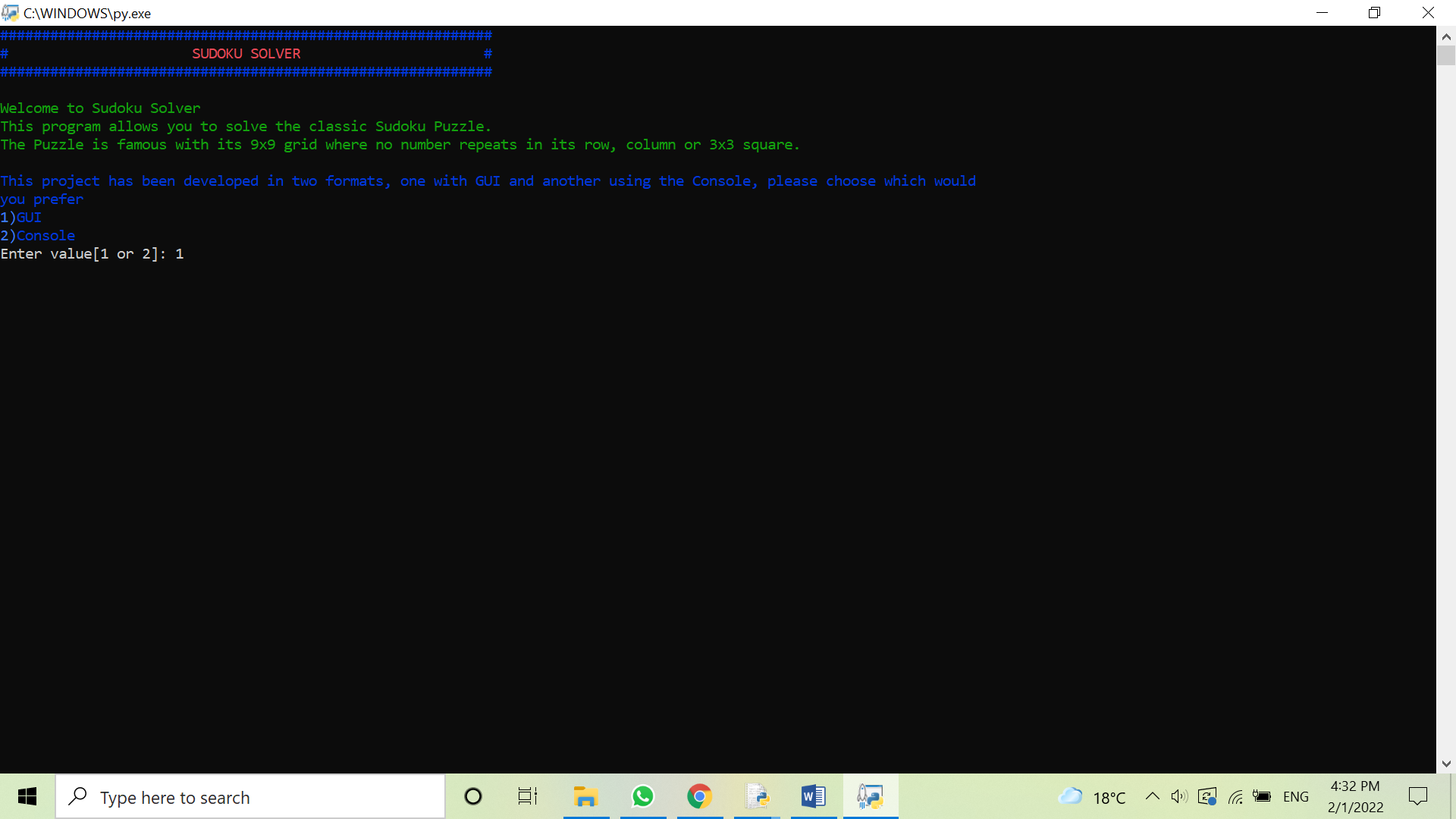
**title.grid(row=0,column=0)**

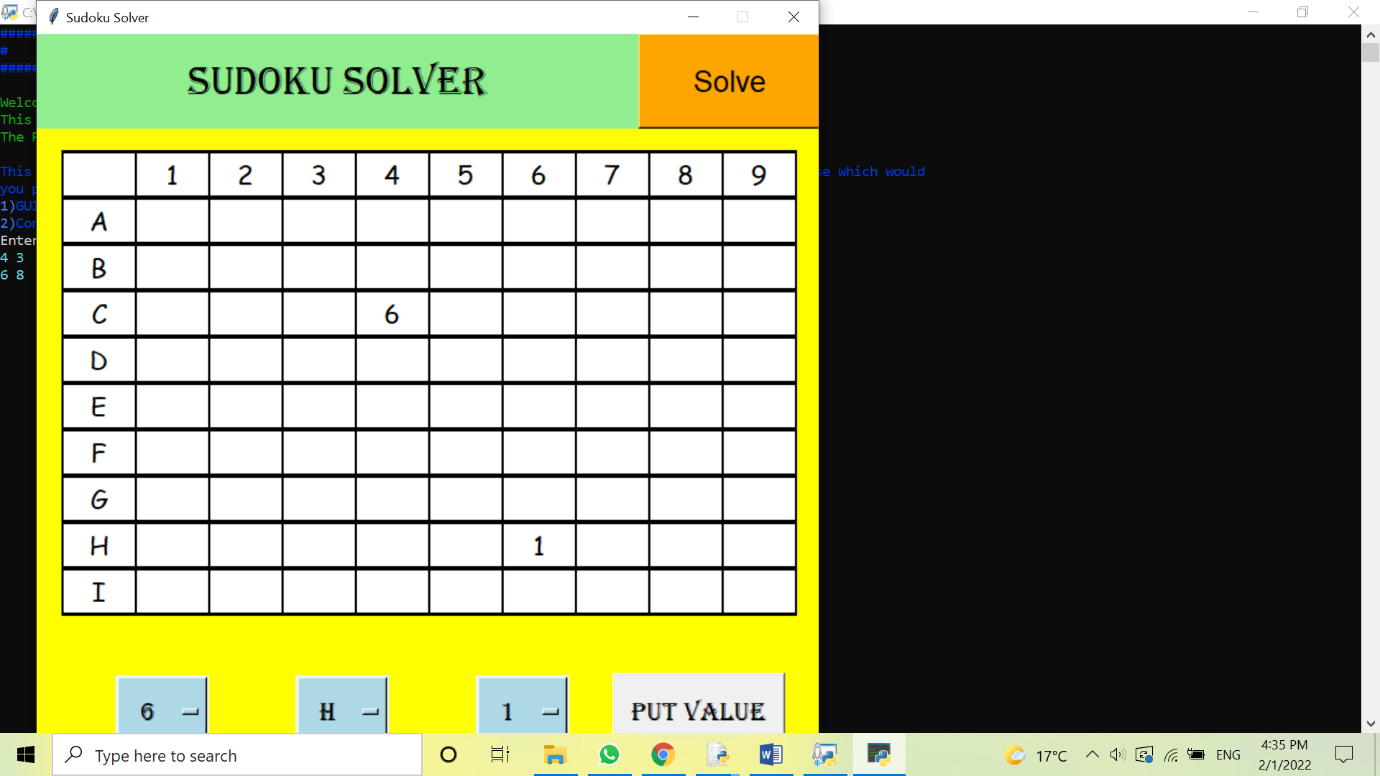
**solveButton.grid(row=0,column=1)**

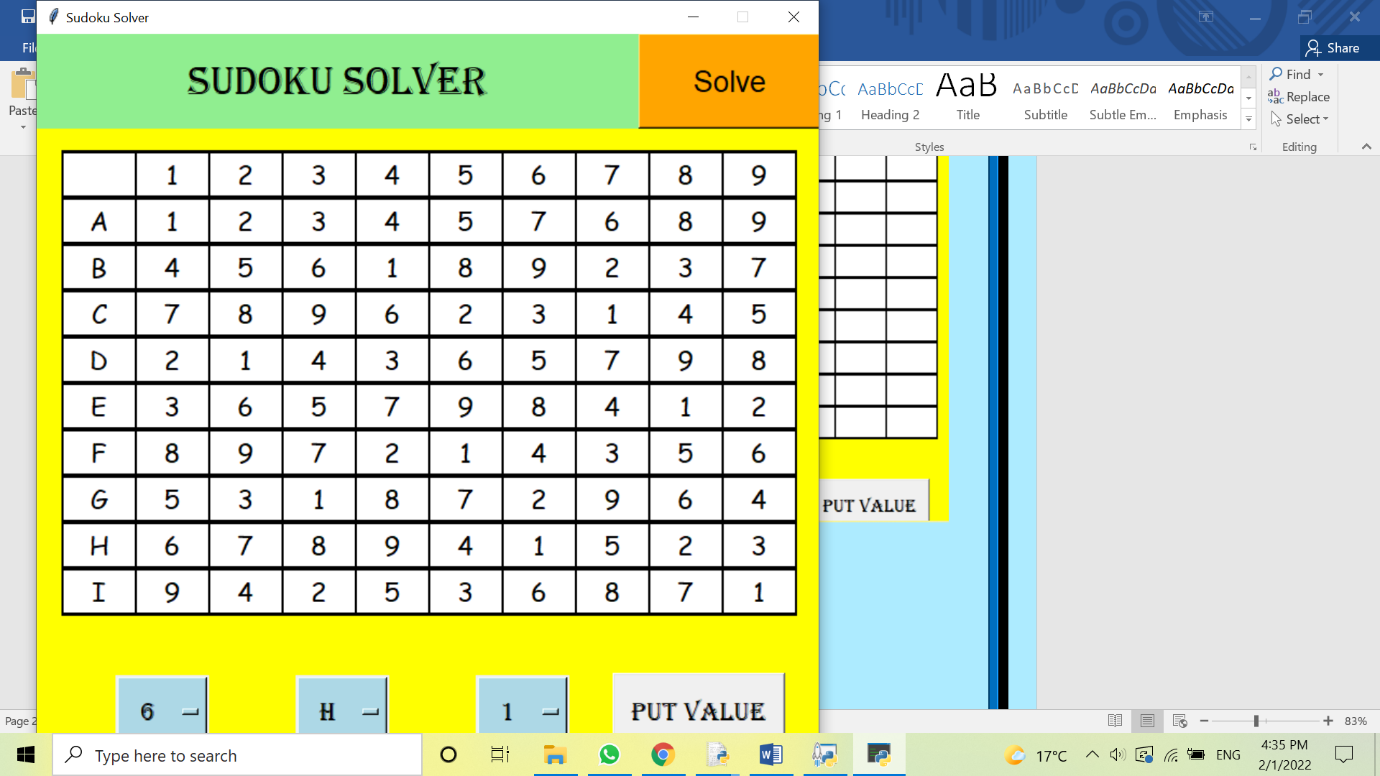
**#begins window launch**

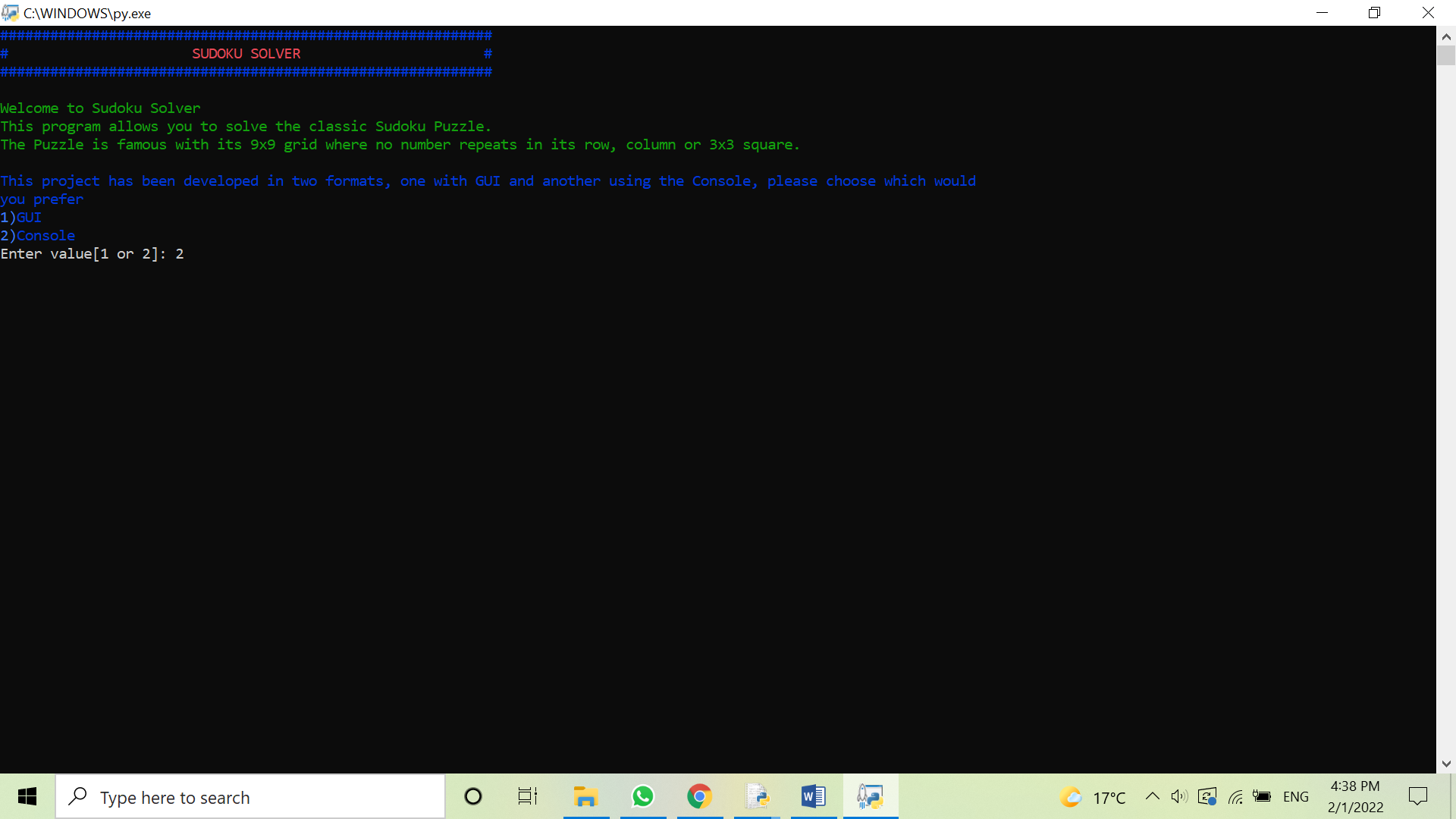
**root.mainloop()**

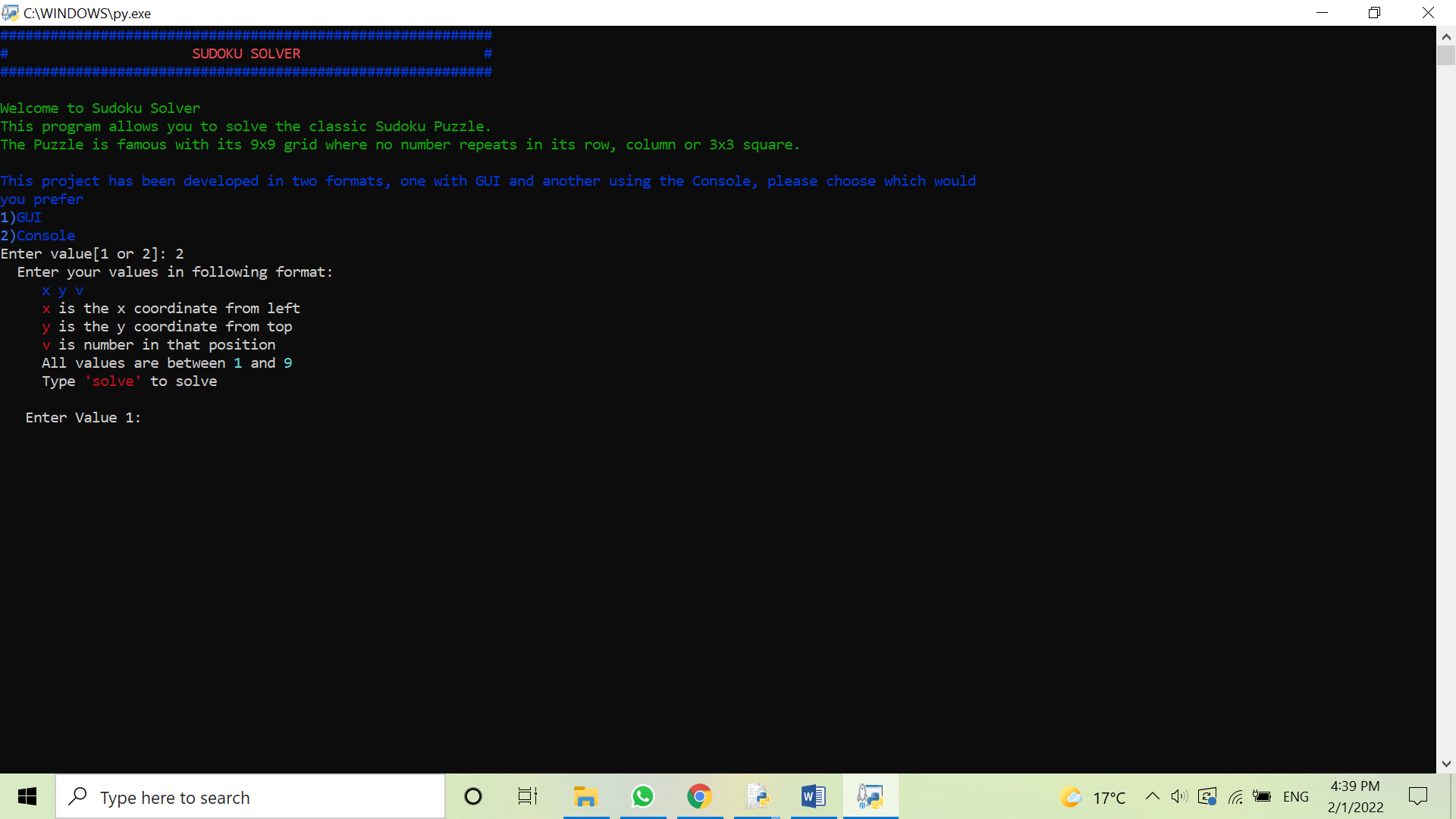
*Output Screens:*

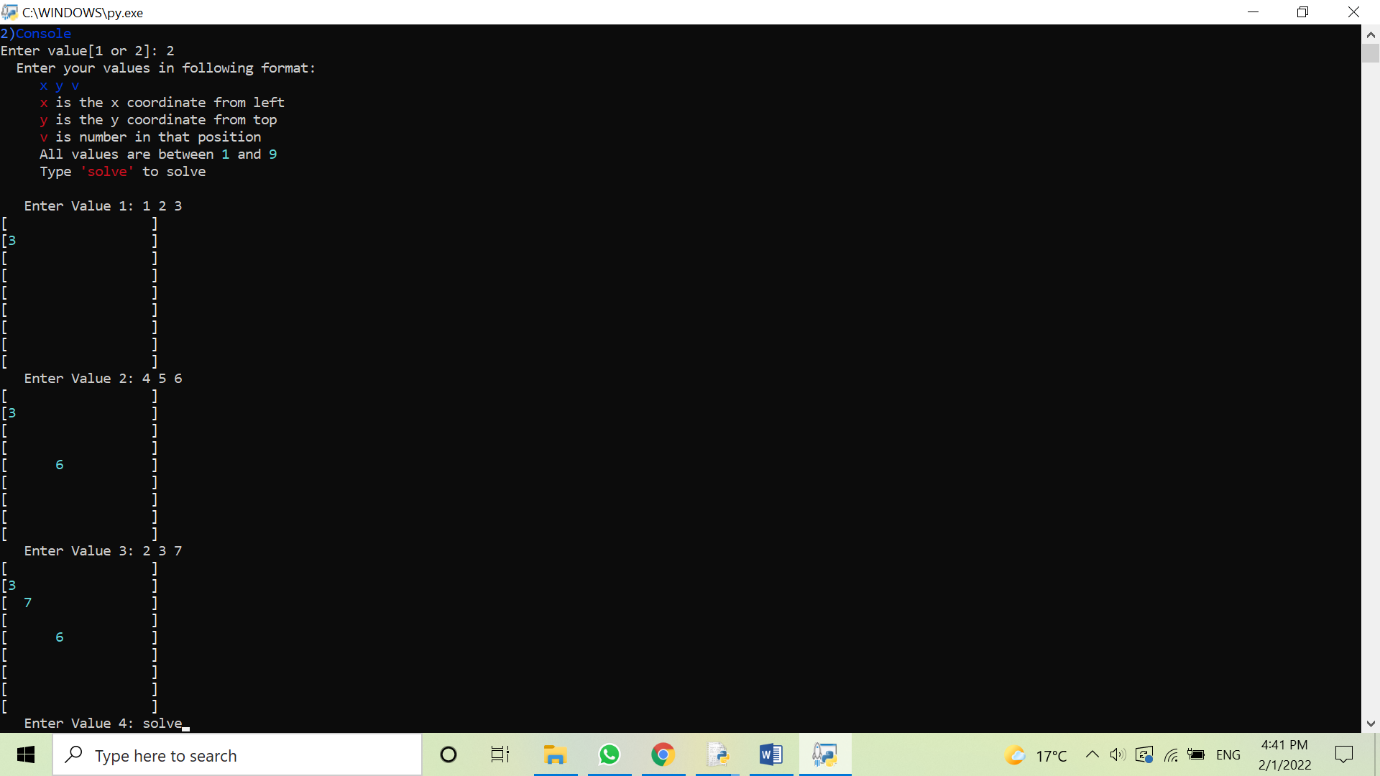


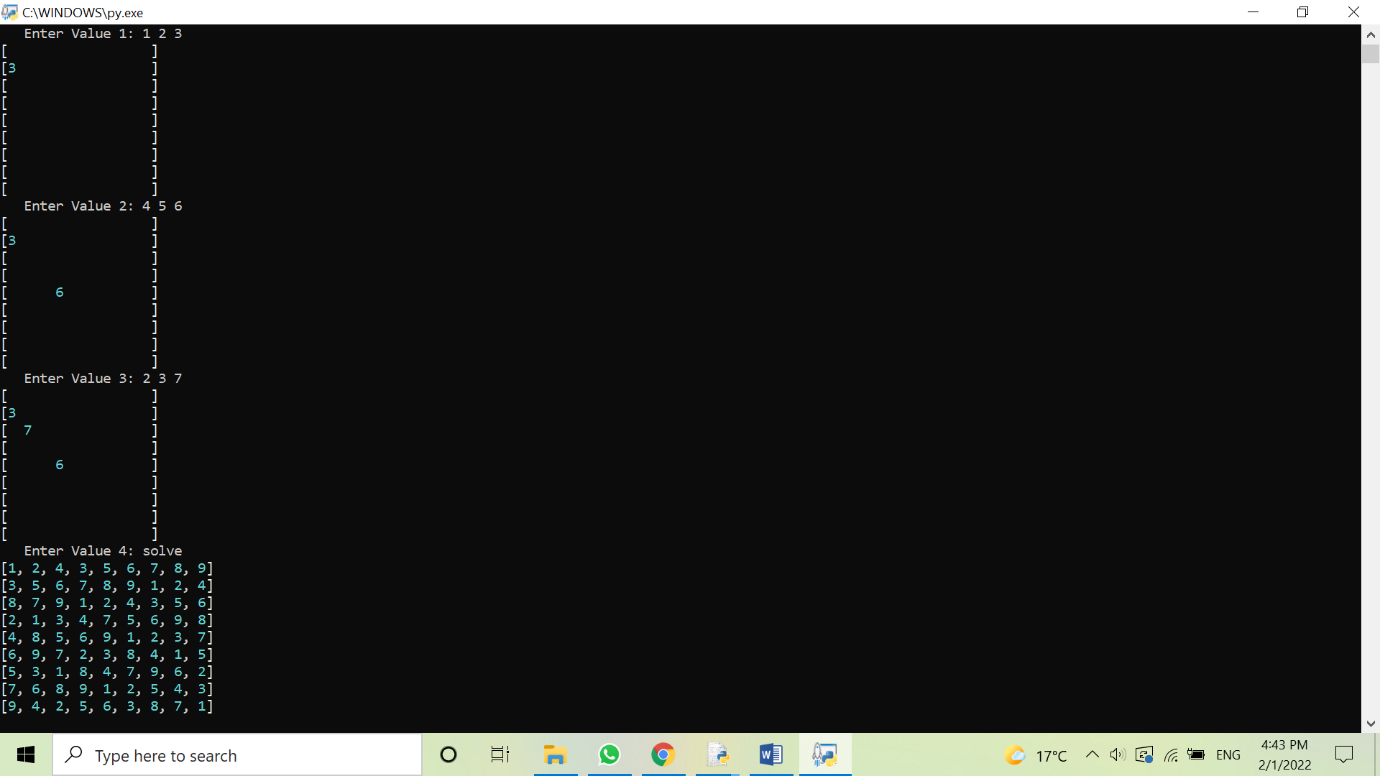












*Future Enhancement of Project:*

In the near future, we would like add a history and random puzzle generator feature to this project.

For the first feature, we would like to add a history tab using which user will be able to see previously solved Sudoku. It can be done using file.

For the second feature, we want to add a system for generating random puzzles that the user can also solve.

*Bibliography:*

* *Text book - Class XI – Computer Science, Preeti Arora*
* *cbse.nic.in*
* [*https://www.geeksforgeeks.org/*](https://www.geeksforgeeks.org/)
* *https://www.w3schools.com/python/*
* [*https://en.wikipedia.org*](https://en.wikipedia.org)