Sudoku Game

Computer Science Project

2023

Abigail Elizabeth Joseph
XII A
Roll No: 1

Index

Acknowledgement

I would like to express my special thanks of gratitude to our principal Ms. Rachel Ignatius and Vice Principal Ms. Preetha Gibu who gave me the opportunity to do this wonderful project on this topic which has helped me to research and learn a lot during the course of completion of this project.
I would also like to thank my family, team members (Anoushka, Jishnu and Nazrin) and Computer Science teacher Mrs.Sreepadma Divakar for constantly encouraging and helping me during this project, which I could not have completed with their support and continuous encouragement.

Introduction

Sudoku is a logic-based, combinatorial number-placement puzzle. In classic Sudoku, the objective is to fill a 9 × 9 grid with digits so that each column, each row, and each of the nine 3 × 3 subgrids that compose the grid contains all of the digits from 1 to 9. The game first appeared in Japan in 1984 where it was given the name “Sudoku,” which is short for a longer expression in Japanese – “Sūji wa dokushin ni kagiru” – which means, “the digits are limited to one occurrence.”

Playing games is something we all do, but it would not be as fun if we didn't have many different levels and some competitiveness in it. Our sudoku game will have different levels to choose from. Aside from the game part, it has time based records of different players and displays the fastest time of each player, the players with the fastest time and the number of sudoku games played by each person. It will also allow a player to change his details like name, age, phone number and email or delete his player profile completely.

The software uses Python 3.12 as front end and Mysql as back end.
It uses the following modules:

- Adding records
- Searching for records and displaying them.
- Updating or editing records
- Deleting records
- Linking the back end and front end

My contributions to this project is linking the back end and front end, searching up records and displaying them, and the functionality of the game as a whole.

Algorithm

Modules Imported

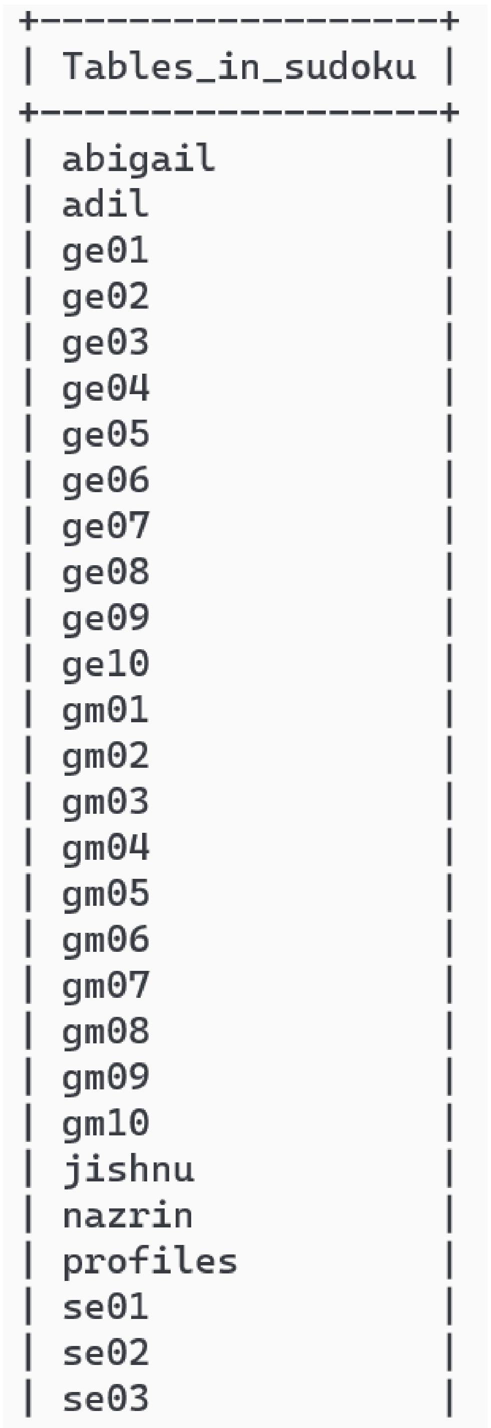
- Mysql connector
- Time module
- Random module
- Pygame
- datetime

User defined functions

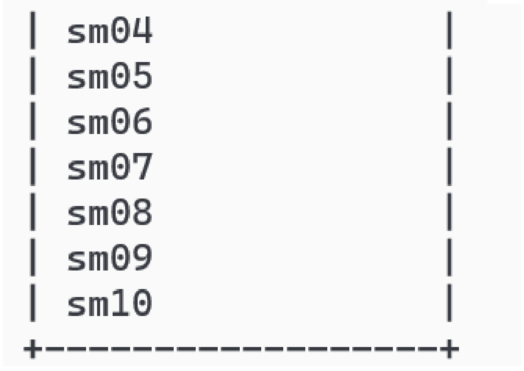
- game_no – generates the game code randomly to play
- check – verifies whether the game is complete and correct. If so ends the game.
- insert – allows the user to insert/remove numbers into the grid.
- sudoku – creates the grid, populates the grid from the mysql tables, and calls the insert function
- account_check – checks whether the user already has an account or not. If not, calls the adding function.
- adding – adds user details inputted into the database creating a new account.
- updation – updates a users/players account details like phone number, age and email.
- delete – deletes a users profile.
- oldest game – displays the date of a users oldest/first game.
- fastest time – displays all of users past games from their fastest time to their slowest time.
- no_of_games – displays the number of games played by a user.

Database Components

Tables in the database:

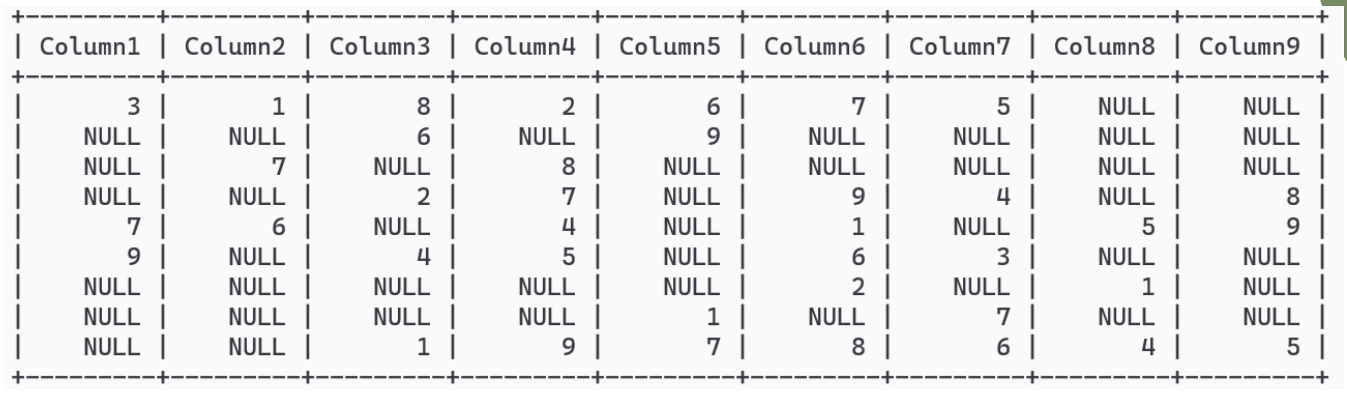


```
+-------------------+
| Tables_in_sudoku  |
+-------------------+
| abigail           |
| adil              |
| ge01              |
| ge02              |
| ge03              |
| ge04              |
| ge05              |
| ge06              |
| ge07              |
| ge08              |
| ge09              |
| ge10              |
| gm01              |
| gm02              |
| gm03              |
| gm04              |
| gm05              |
| gm06              |
| gm07              |
| gm08              |
| gm09              |
| gm10              |
| jishnu            |
| nazrin            |
| profiles          |
| se01              |
| se02              |
| se03              |
```



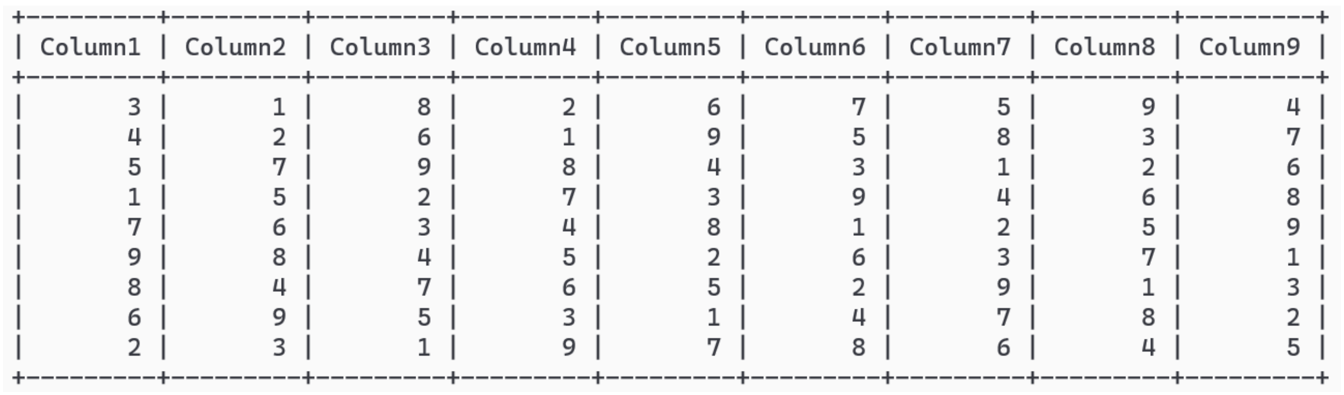
```
| sm04              |
| sm05              |
| sm06              |
| sm07              |
| sm08              |
| sm09              |
| sm10              |
+-------------------+
```

GE01 table:



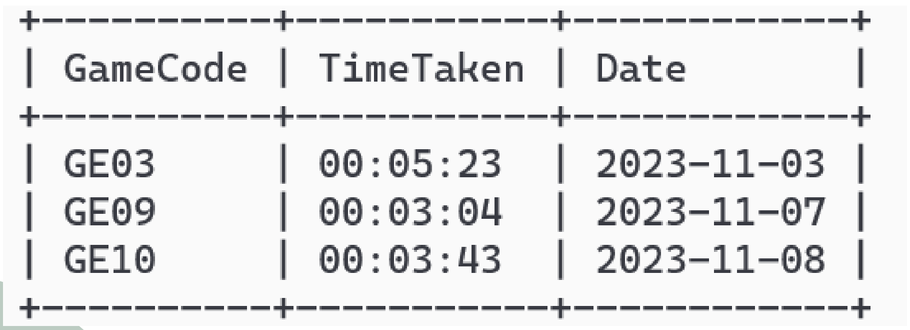
```
+---------+---------+---------+---------+---------+---------+---------+---------+---------+
| Column1 | Column2 | Column3 | Column4 | Column5 | Column6 | Column7 | Column8 | Column9 |
+---------+---------+---------+---------+---------+---------+---------+---------+---------+
|       3 |       1 |       8 |       2 |       6 |       7 |       5 |    NULL |    NULL |
|    NULL |    NULL |       6 |    NULL |       9 |    NULL |    NULL |    NULL |    NULL |
|    NULL |       7 |    NULL |       8 |    NULL |    NULL |    NULL |    NULL |    NULL |
|    NULL |    NULL |       2 |       7 |    NULL |       9 |       4 |    NULL |       8 |
|       7 |       6 |    NULL |       4 |    NULL |       1 |    NULL |       5 |       9 |
|       9 |    NULL |       4 |       5 |    NULL |       6 |       3 |    NULL |    NULL |
|    NULL |    NULL |    NULL |    NULL |    NULL |       2 |    NULL |       1 |    NULL |
|    NULL |    NULL |    NULL |    NULL |       1 |    NULL |       7 |    NULL |    NULL |
|    NULL |    NULL |       1 |       9 |       7 |       8 |       6 |       4 |       5 |
+---------+---------+---------+---------+---------+---------+---------+---------+---------+
```

SE01 table:



```
+---------+---------+---------+---------+---------+---------+---------+---------+---------+
| Column1 | Column2 | Column3 | Column4 | Column5 | Column6 | Column7 | Column8 | Column9 |
+---------+---------+---------+---------+---------+---------+---------+---------+---------+
|       3 |       1 |       8 |       2 |       6 |       7 |       5 |       9 |       4 |
|       4 |       2 |       6 |       1 |       9 |       5 |       8 |       3 |       7 |
|       5 |       7 |       9 |       8 |       4 |       3 |       1 |       2 |       6 |
|       1 |       5 |       2 |       7 |       3 |       9 |       4 |       6 |       8 |
|       7 |       6 |       3 |       4 |       8 |       1 |       2 |       5 |       9 |
|       9 |       8 |       4 |       5 |       2 |       6 |       3 |       7 |       1 |
|       8 |       4 |       7 |       6 |       5 |       2 |       9 |       1 |       3 |
|       6 |       9 |       5 |       3 |       1 |       4 |       7 |       8 |       2 |
|       2 |       3 |       1 |       9 |       7 |       8 |       6 |       4 |       5 |
+---------+---------+---------+---------+---------+---------+---------+---------+---------+
```

Abigail table:



```
+----------+-----------+------------+
| GameCode | TimeTaken | Date       |
+----------+-----------+------------+
| GE03     | 00:05:23  | 2023-11-03 |
| GE09     | 00:03:04  | 2023-11-07 |
| GE10     | 00:03:43  | 2023-11-08 |
+----------+-----------+------------+
```

Code

```
import mysql.connector as mysql
import time
import random
import pygame
from datetime import datetime
mycon=mysql.connect(host="localhost",user="root",pass
word="8921980499" ,database="Sudoku")
mycur=mycon.cursor()
original_grid_element_colour=(52,31,151)
width=550
background_colour=(251,247,245)
#grid and game
def game_no():
    print("Easy, Medium or Hard")
    level=input("Enter difficulty level:")
    a=random.randint(1,10)
    num=str(a)
    if len(num)==1:
        num="0"+str(a)
    if level.upper()=="EASY":
        game="E"
    elif level.upper()=="MEDIUM":
        game="M"
    elif level.upper()=="HARD":
        game="H"
    game_code="G"+game+num
    solved_code="S"+game+num
    return (game_code,solved_code)
def check():
    mycur.execute("select * from %s"%(game[1],))
    solved=mycur.fetchall()
    lsolved=[]
    for i in range(len(solved)):
        lsolved.append(list(solved[i]))
    count=0
    for row in range(len(lsolved)):
        if lsolved[row]==grid_check[row]:
            count+=1
    if count==9:
        print("Yay correct. Game over.")
        stop_time=time.time()
        global time_taken
        time_taken=round(stop_time-start_time,0)
        return "correct"
    return
def insert(win,position):
    i,j = position[1], position[0]
    myfont = pygame.font.SysFont("monospace", 35)
    buffer=5
    grid_original = [[grid[x][y] for y in range(len(grid[0]))] for
x in range(len(grid))]
    while True:
        for event in pygame.event.get():
            if event.type == pygame.QUIT:
                return
            if event.type == pygame.KEYDOWN:
                if(grid_original[i-1][j-1] != None):
```

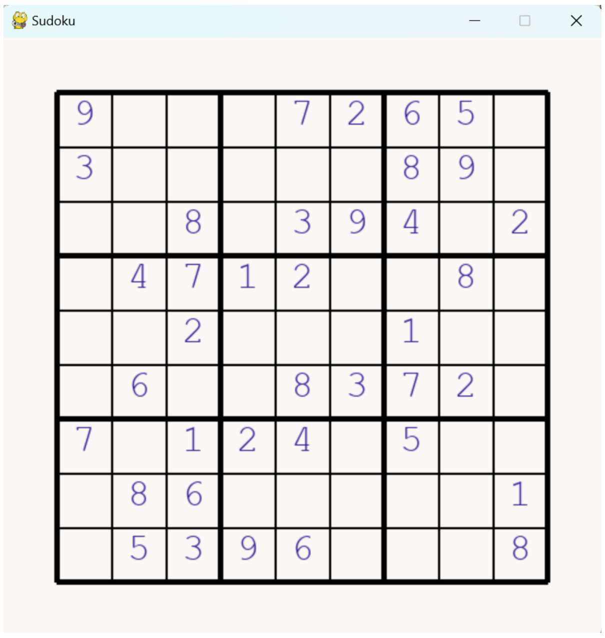
```
                return
            if(event.key == 48): #checking with 0
                grid_original[i-1][j-1] = event.key - 48
                grid_check[i-1][j-1] = event.key - 48
                pygame.draw.rect(win, background_colour,
(position[0]*50 + buffer, position[1]*50+ buffer,50
-2*buffer , 50 - 2*buffer))
                pygame.display.update()
                return
            if(0 < event.key - 48 <10):  #We are checking for
valid input
                pygame.draw.rect(win, background_colour,
(position[0]*50 + buffer, position[1]*50+ buffer,50
-2*buffer , 50 - 2*buffer))
                value = myfont.render(str(event.key-48), True,
(0,0,0))
                win.blit(value, (position[0]*50 +15,
position[1]*50))
                grid_original[i-1][j-1] = event.key - 48  #grid
inputs the numbers, event key -48 gives the number to be
inputed
                grid_check[i-1][j-1] = event.key - 48
                pygame.display.update()
                return
def sudoku():
    pygame.init()
    win=pygame.display.set_mode((width,width))
    pygame.display.set_caption("Sudoku")
    win.fill(background_colour)
    global myfont
    myfont=pygame.font.SysFont("monospace",35)
    for x in range(0,10):
        if (x%3==0):
            pygame.draw.line(win,(0,0,0),(50+50*x,50),
(50+50*x,500),5)
            pygame.draw.line(win,(0,0,0),(50,50+50*x),
(500,50+50*x),5)
        pygame.draw.line(win,(0,0,0),(50+50*x,50),
(50+50*x,500),2)
        pygame.draw.line(win,(0,0,0),(50,50+50*x),
(500,50+50*x),2)
    font=pygame.font.SysFont('pixeltype.ttf',35)
    pygame.display.update()
    global game
    game=game_no()
    mycur.execute("select * from %s"%(game[0],))
    global grid
    grid=mycur.fetchall()
    global grid_check
    grid_check=[[grid[x][y] for y in range(len(grid[0]))] for x
in range(len(grid))]
    for a in range(0,len(grid[0])):
        for b in range(0,len(grid[0])):
            if grid[a][b]==None:
                continue
            else:
                if(0<grid[a][b]<10):
                    value=myfont.render(str(grid[a]
[b]),True,original_grid_element_colour)
                    win.blit(value,((b+1)*50+15,(a+1)*50))
    pygame.display.update()
    global start_time
    start_time=time.time()
    #print(start_time)
    while True:
        for event in pygame.event.get():
            if event.type == pygame.MOUSEBUTTONDOWN and
event.button == 1:
                pos = pygame.mouse.get_pos()
                insert(win,(pos[0]//50,pos[1]//50))
                correct=check()
                if correct=="correct":
                    pygame.quit()
                    return
            if event.type == pygame.QUIT:
                pygame.quit()
                return
#backend work
def account_check():
    existing_player=input("Do you have an account?(y/n)")
    if existing_player.upper()=="N":
        name = input("Enter name: ")
        age = int(input("Enter age: "))
        phoneno = input("Enter Phone Number: ")
        gender = input("Enter Gender(M/F): ")
        mycur.execute("select max(PlayerId) from profiles")
        lmax_playerid=mycur.fetchall()
        max_playerid=lmax_playerid[0][0]
        global playerid
        playerid=max_playerid + 1
        insert_query = "INSERT INTO profiles
(PlayerId,PlayerName,PhoneNo,Age,Gender) VALUES
({},'{}',{},
{},'{}')".format(playerid,name,phoneno,age,gender)
        mycur.execute(insert_query)
        mycur.execute("create table {} like
Abigail".format(name))
        mycon.commit()
        print("Added!")
        print("Your player id is",playerid)
    elif existing_player.upper()=="Y":
        playerid=int(input("Enter your player id:"))
        return
def adding():
    mycur.execute("select PlayerName from profiles where
PlayerId=%s"%(playerid,))
    player_table=mycur.fetchall()[0][0]
    insert_query="INSERT INTO "+player_table+" VALUES
(%s, %s, curdate())"
    values=(game[0],time_taken)
    mycur.execute(insert_query,values)
    mycon.commit()
    return
def updation():
    def update_phoneno(player_id, new_phone_number):
        update_query = "UPDATE profiles SET PhoneNo = %s
WHERE PlayerId = %s"
        inp= (new_phone_number, player_id)
        mycur.execute(update_query, inp)
        return
    def update_age(player_id, new_age):
        update_query = "UPDATE profiles SET Age = %s
WHERE PlayerId = %s"
        inp = (new_age,player_id)
        mycur.execute(update_query,inp)
        return
    def update_gender(player_id,gender):
        update_query = "UPDATE profiles SET Gender = %s
WHERE PlayerId = %s"
        inp = (gender,player_id)
        mycur.execute(update_query,inp)
        return
    while True:
        print("1. Update Phone Number")
        print("2. Update Age")
        print("3. Update gender")
        print("4. Quit")
        ch = int(input("Enter your choice: "))
        if ch == 1:
            player_id = int(input("Enter player ID: "))
            new_phoneno = input("Enter new phone number: ")
            update_phoneno(player_id, new_phoneno)
            mycon.commit()
            print("Phone number updated .")
        elif ch == 2:
            player_id = int(input("Enter player ID: "))
            new_age = int(input("Enter new age: "))
            update_age(player_id, new_age)
            mycon.commit()
            print("Age updated .")
        elif ch==3:
            player_id = int(input("Enter player ID: "))
            gender=input("Enter new gender: ")
            update_gender(player_id, gender)
            mycon.commit()
        elif ch == 4:
            break
        else:
            print("Invalid choice")
    return
def delete():
    player_id=input("Enter the ID of the player whose
records you want to delete:")
    delete_query="DELETE FROM PROFILES WHERE
PLAYERID=%s"
    data=(player_id,)#tuple containing id to delete
    mycur.execute(delete_query,data)
    print("The profile of player",player_id,"has been
deleted")
    mycon.commit()
    return
def oldest_game():
    mycur.execute("select PlayerName from profiles where
PlayerId=%s"%(playerid,))
    player_table=mycur.fetchall()[0][0]
    mycur.execute("select min(Date) from  "+player_table)
    oldest=mycur.fetchall()
    print("Your oldest/first game was played on",oldest[0]
[0].strftime("%d-%b-%Y"))
    return
def fastest_times():
    mycur.execute("select PlayerName from profiles where
PlayerId=%s"%(playerid,))
    player_table=mycur.fetchall()[0][0]
    mycur.execute("select * from "+player_table+" order by
timetaken")
    times=mycur.fetchall()
    col_names=[i[0] for i in mycur.description]
    for name in col_names:
        print(str(name).center(10),"\t",end="")
    print()
    for i in times:
        for j in i:
            print(str(j).center(10),"\t",end="")
        print()
    return
def no_of_games():
    mycur.execute("select PlayerName from profiles where
PlayerId=%s"%(playerid,))
    player_table=mycur.fetchall()[0][0]
    mycur.execute("select count(GameCode) from
"+player_table)
    count=mycur.fetchall()[0][0]
    print("You have played",count,"sudoku games,")
#main code
account_check()
print("1. Play Game")
print("2. How to play Sudoku")
print("3. See a players oldest game")
print("4. See a players fastest times")
print("5. See number of games played")
print("6. Update player profile")
print("7. Delete a players proflie completely")
print("8. Quit")
while True:
    opt=int(input("Enter your option:"))
    if opt==1:
        sudoku()
        adding()
    elif opt==2:
        print('''Traditional Sudoku is a 9x9 puzzle grid made
up of nin 3x3 regions. What you need to do is to complete
the Sudoku puzzle and make sure that the same single
number may not appear twice in the same row, column, or
any of the nine 3x3 regions.''')
    elif opt==3:
        oldest_game()
    elif opt==4:
        fastest_times()
    elif opt==5:
        no_of_games()
    elif opt==6:
        updation()
    elif opt==7:
        delete()
        break
```

```
elif opt==8:
    break
else:
    print("Invalid option")
```

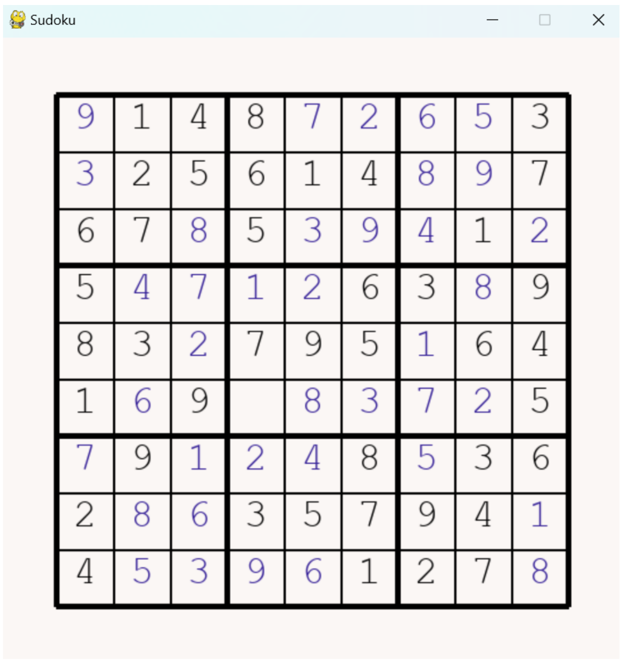
Output

```
pygame 2.5.2 (SDL 2.28.3, Python 3.12.0)
Hello from the pygame community. https://www.pygame.org/contribute.html
Do you have an account?(y/n)n
Enter name: John
Enter age: 17
Enter Phone Number: 91873463463
Enter Gender(M/F): M
Added!
Your player id is 104
1. Play Game
2. How to play Sudoku
3. See a players oldest game
4. See a players fastest times
5. See number of games played
6. Update player profile
7. Delete a players proflie completely
8. Quit
Enter your option:2
Traditional Sudoku is a 9x9 puzzle grid made up of nin 3x3 regions. What you need to do is to c
omplete the Sudoku puzzle and make sure that the same single number may not appear twice in the
same row, column, or any of the nine 3x3 regions.
Enter your option:1
Easy, Medium or Hard
Enter difficulty level:easy
Yay correct. Game over.
Enter your option:3
Your oldest/first game was played on 15-Nov-2023
Enter your option:4
 GameCode       TimeTaken          Date
   GE06          0:03:18       2023-11-15
```

```
Enter your option:6
1. Update Phone Number
2. Update Age
3. Update gender
4. Quit
Enter your choice: 1
Enter player ID: 104
Enter new phone number: 93475136843
Phone number updated .
1. Update Phone Number
2. Update Age
3. Update gender
4. Quit
Enter your choice: 4
Enter your option:5
You have played 1 sudoku games,
Enter your option:7
Enter the ID of the player whose records you want to delete:104
The profile of player 104 has been deleted
```



9				7	2	6	5	
3						8	9	
		8		3	9	4		2
	4	7	1	2			8	
		2				1		
	6			8	3	7	2	
7		1	2	4		5		
	8	6						1
	5	3	9	6				8



9	1	4	8	7	2	6	5	3
3	2	5	6	1	4	8	9	7
6	7	8	5	3	9	4	1	2
5	4	7	1	2	6	3	8	9
8	3	2	7	9	5	1	6	4
1	6	9		8	3	7	2	5
7	9	1	2	4	8	5	3	6
2	8	6	3	5	7	9	4	1
4	5	3	9	6	1	2	7	8

Bibliography

- https://www.pygame.org/docs/
- https://youtu.be/FfWpgLFMI7w?si=KkxTx-SM-mxBWkva
- https://youtu.be/I2lOwRiGNy4?si=gSP5La64heqg5ja1
- SUMITA AURORA - Computer Science in Python Textbook from Class XII