**THE MULTI GAME PROGRAM**

## Contains 2 Games - 2048 Game and Tic Tac Toe

NAME :

ADMISSION NO. :

CLASS :

ROLL NO. :

SUBJECT : COMPUTER SCIENCE

(PYTHON) (CODE: 083)

SCHOOL : DELHI PUBLIC SCHOOL SURAT

ACADEMIC YEAR : 2020-21

**INDEX**

| ***SR NO.*** | ***CONTENTS*** | ***PAGE NO.*** |
| --- | --- | --- |
| 1. | ACKNOWLEDGEMENT | 3 |
| 2. | CERTIFICATE | 4 |
| 3. | INTRODUCTION | 5 |
| 4. | EXTERNAL MODULES USED | 6,7 |
| 5. | CODING | 8-27 |
| 6. | OUTPUT | 28-57 |

**INTRODUCTION**

This is a Class 12 Computer Science project which has SQL Connectivity in Python.We have made a multi game application with 2 games - 2048 game and Tic Tac Toe game and have used SQL Connectivity to store user data in a SQL database. A SQL table contains records of all the users.We have also used the Pygame module to create and display these games.

The application starts with displaying a page where the user can enter their username and once they have done that, a page to enter the password is displayed. Then, the homepage is displayed which contains different options where the players can press certain keys to access either of the two games, the leaderboard or their user profile.

The user profile displays the user data stored in the SQL database which includes the player number, username, password, and points. There is also a provision to delete the user record which will permanently delete the record of the user from the SQL table. This provision can be accessed only when the user is on their user profile page.

Whenever the player plays either of the games they can earn a certain number of points and these points are saved in their record in the SQL table which is updated every time the user gets more points.

The leaderboard displays the top ten players with the highest points and is used to keep a track of the best players.

In conclusion, our application is a class 12 project which demonstrates how some of the concepts we have learnt in class 11th and 12th can be applied .

**EXTERNAL MODULES USED**

1. random -

* random.choice()
* random.randrange()

1. pygame -

* pygame.init()
* pygame.display.set\_mode()
* pygame.display.set\_caption()
* pygame.event.get()
* pygame.display.update()
* pygame.time.Clock()
* pygame.draw.rect()
* <display object used>.fill(<tuple of the colour used>)
* <display object used>.blit()
* pygame.display.update()
* pygame.font.Sysfont(<font type>,<font size>)
* pygame.image.load()
* pygame.transform.scale()
* pygame.event.poll()
* pygame.key.get\_pressed()
* pygame.key.name(event.key)
* pygame.event.get()
* event.type
* event.key
* pygame.QUIT
* pygame.KEYDOWN
* pygame.K\_SPACE
* pygame.K\_BACKSPACE
* pygame.K\_RETURN
* pygame.K\_1
* pygame.K\_2
* pygame.K\_3
* pygame.K\_4
* pygame.K\_5
* pygame.K\_6
* pygame.K\_7
* pygame.K\_8
* pygame.K\_9
* pygame.K\_LSHIFT
* pygame.K\_RSHIFT
* pygame.K\_LEFT
* pygame.K\_RIGHT
* pygame.K\_UP
* pygame.K\_DOWN
* pygame.draw.line(<display object>,<colour>,<coordinates of starting point>,<coordinates of ending point>,<width>)
* <font object>.render(<text to add>,<colour of text>)
* pygame.quit()

1. mysql.connector -

* mysql.connector.connect()
* <connection object>.cursor()
* <cursor object>.execute()
* <connection object used>.commit()
* <cursor object>.fetchall()
* <cursor object>.fetchone()
* <cursor object>.close()
* <connection object>.close()

**CODE**

import pygame

import random

#function definitions and classes for 2048 game

class Main:

def \_\_init\_\_(self):

# SETUP pygame environment

pygame.init()

self.screen = pygame.display.set\_mode((500,500))

pygame.display.set\_caption('2048')

self.clock = pygame.time.Clock()

# Create the Board

self.board = Board()

#self.board.add\_tile()

self.run()

def paint(self):

# Paint the background

self.screen.fill((160,223,254))

# Write that by pressing Enter can return to home screen

global tse

self.screen.blit(tse,(80,460))

# Paint the board object to screen

self.board.paint(self.screen)

pygame.display.update()

# Handling pygame.QUIT event, and

# KEYDOWNS. Keydowns is parsed to board.move function, Return for exiting game and going to home page

def eventhandler(self):

global v

for event in pygame.event.get():

if event.type == pygame.QUIT:

global k

global z

global rk

self.running = False

v=k=z=False

rk=0

if event.type == pygame.KEYDOWN:

if event.key == pygame.K\_LEFT:

self.board.move("LEFT")

if event.key == pygame.K\_RIGHT:

self.board.move("RIGHT")

if event.key == pygame.K\_UP:

self.board.move("UP")

if event.key == pygame.K\_DOWN:

self.board.move("DOWN")

if event.key==pygame.K\_RETURN:

self.running=False

v=False

# Update function.

def update(self):

self.board.update\_tiles()

# Main loop. 60 FPS

def run(self):

self.running = True

while self.running:

self.clock.tick(60)

self.eventhandler()

self.update()

self.paint()

class Tile:

def \_\_init\_\_(self,x,y,stage):

# SETUP tiles x,y and stage. Stage is the number it represents, 2,4,8 etc.

self.x = x

self.y = y

self.stage = stage

self.colorlist = [(245,240,255),(108,187,244),(177,200,241),(147,164,247),(151,157,232),(158,189,224),(123,183,227),(126,145,207),(126,157,190),(184,154,245),(129,203,193)]

# Move the tile to new x,y coordinates. Returns False if it moves into a wall.

def move\_tile(self,x=0,y=0):

self.x += x

self.y += y

if self.x<0 or self.x > 3 or self.y < 0 or self.y > 3:

self.x -= x

self.y -=y

return False

return True

# Merge two tiles.

def merge(self,Tile):

global p

if Tile.stage == self.stage:

self.increasestage()

p+=1 #increasing points of user if they add two tiles

return True

else:

return False

def increasestage(self):

self.stage += 1

# Draw the tile to Board.

def draw(self,screen,x,y,font):

pygame.draw.rect(screen,self.colorlist[self.stage-1],(x,y,87,87))

# draw the numbers on tiles:

if self.stage <= 2:

color = (120,110,101)

else:

color = (250,248,239)

text = font.render(str(2\*\*self.stage),2,color)

screen.blit(text,(x+(87//2 - text.get\_width()//2), y + (87//2 -text.get\_height()//2)))

class Board:

def \_\_init\_\_(self):

# self.tiles keep track of the tiles GUI positions.

self.tiles = [[0,0,0,0] for i in range(4)]

self.board = pygame.Rect(50,50,400,400)

self.color = (2,170,251)

# tilearray stores the tiles as a list. When self.update\_tiles is called

# the tiles in tile\_array gets updated in self.tiles (the tiles GUI position)

self.tilearray = []

self.add\_tile()

self.add\_tile()

self.font = pygame.font.SysFont('comicsans',61)

#Draw the board background to screen.

def paint(self,screen):

pygame.draw.rect(screen,self.color,self.board)

self.drawtiles(screen)

# Draw tiles to screen. If no tile, draw empty square.

def drawtiles(self,screen):

for i,array in enumerate(self.tiles):

for j,tile in enumerate(array):

if tile == 0:

pygame.draw.rect(screen,(204,193,180),(60+i\*87+10\*i,60+j\*87+10\*j,87,87))

else:

tile.draw(screen,60+i\*87+10\*i,60+j\*87+10\*j,self.font)

# Returns an arraylist with positions in self.tiles which are empty

def get\_empty\_spaces(self):

empty = []

for i,array in enumerate(self.tiles):

for j,tile in enumerate(array):

if tile==0:

empty.append([i,j])

return empty

# Add a new tile to the game. Coordinates chosen at random.

def add\_tile(self):

empty = self.get\_empty\_spaces()

chosen = random.choice(empty)

if random.randrange(1,100) <10:

stage = 2

else:

stage = 1

t = Tile(chosen[0],chosen[1],stage)

self.tilearray.append(t)

# Move all tiles on the board.

def move(self,key):

stepstaken = 0

if key=="LEFT":

for i, array in enumerate(self.tiles):

for j, \_ in enumerate(array):

tile = self.tiles[j][i]

if tile!=0:

stepstaken += self.move\_single\_tile(tile,-1,0)

self.update\_tiles()

if key =="RIGTH":

for i,array in enumerate(self.tiles):

for j,\_ in enumerate(array):

tile = self.tiles[3-j][3-i]

if tile!= 0:

stepstaken += self.move\_single\_tile(tile,1,0)

self.update\_tiles()

if key == "UP":

for i,array in enumerate(self.tiles):

for j,\_ in enumerate(array):

tile = self.tiles[i][j]

if tile!=0:

stepstaken += self.move\_single\_tile(tile,0,-1)

self.update\_tiles()

if key == "DOWN":

for i, array in enumerate(self.tiles):

for j,\_ in enumerate(array):

tile = self.tiles[3-i][3-j]

if tile!=0:

stepstaken += self.move\_single\_tile(tile,0,1)

self.update\_tiles()

if stepstaken>0:

self.add\_tile()

# Tiles are stored in self.tilearray. When updating, the tiles from self.tilearray is

# stored in the 2d array.

def move\_single\_tile(self,tile,vx=0,vy=0):

steps = 0

for i in range(0,3):

if self.position\_is\_inside\_grid(tile.x+vx,tile.y+vy) and self.tile\_is\_empty(tile.x+vx,tile.y+vy):

tile.move\_tile(vx,vy)

steps+=1

else:

if self.position\_is\_inside\_grid(tile.x+vx,tile.y+vy) and self.tiles[tile.x+vx][tile.y+vy].merge(tile):

self.tilearray.remove(tile)

steps += 1

return steps

def position\_is\_inside\_grid(self,x,y):

if x>-1 and x<4 and y>-1 and y<4:

return True

else:

return False

def tile\_is\_empty(self,x,y):

if self.tiles[x][y] == 0:

return True

else:

return False

def update\_tiles(self):

self.tiles = [[0,0,0,0] for i in range(4)]

for tile in self.tilearray:

self.tiles[tile.x][tile.y] = tile

pygame.init()

dis=pygame.display.set\_mode((600,600)) #display

t=pygame.time.Clock() #clock

fo=pygame.font.SysFont('Palatino Linotype',20) #font

#loading and resizing image of tic tac toe, user profile, leaderboard, login again and point system for home page

TTTim=pygame.image.load("C:/Users/admin/OneDrive/Pictures/Screenshots/2048 tic tac toe.jpg")

TTTim=pygame.transform.scale(TTTim,(350,300))

Upim=pygame.image.load("C:/Users/admin/OneDrive/Pictures/Screenshots/user profile.jpeg")

Upim=pygame.transform.scale(Upim,(265,100))

Lbim=pygame.image.load("C:/Users/admin/OneDrive/Pictures/Screenshots/leaderboard.jpeg")

Lbim=pygame.transform.scale(Lbim,(265,100))

Ptim=pygame.image.load("C:/Users/admin/OneDrive/Pictures/Screenshots/point system.jpeg")

Ptim=pygame.transform.scale(Ptim,(550,140))

Luim=pygame.image.load("C:/Users/admin/OneDrive/Pictures/Screenshots/different login.jpg")

Luim=pygame.transform.scale(Luim,(190,300))

#loading and resizing images of 'X' and 'O' for tic tac toe

Xim=pygame.image.load("C:/Users/admin/OneDrive/Pictures/Screenshots/x.jpg")

Oim=pygame.image.load("C:/Users/admin/OneDrive/Pictures/Screenshots/o.jpg")

Xim=pygame.transform.scale(Xim,(175,175))

Oim=pygame.transform.scale(Oim,(175,175))

#loading and resizing images to declare the winner or draw for tic tac toe

Pxim=pygame.image.load("C:/Users/admin/OneDrive/Pictures/Screenshots/player 1 won.jpg")

Poim=pygame.image.load("C:/Users/admin/OneDrive/Pictures/Screenshots/player 2 won.jpg")

Pdim=pygame.image.load("C:/Users/admin/OneDrive/Pictures/Screenshots/draw.jpg")

Pxim=pygame.transform.scale(Pxim,(500,400))

Poim=pygame.transform.scale(Poim,(500,400))

Pdim=pygame.transform.scale(Pdim,(500,400))

#loading and resizing images of the numbers of the boxes for tic tac toe

im1=pygame.image.load("C:/Users/admin/OneDrive/Pictures/Screenshots/1.jpg")

im2=pygame.image.load("C:/Users/admin/OneDrive/Pictures/Screenshots/2.jpg")

im3=pygame.image.load("C:/Users/admin/OneDrive/Pictures/Screenshots/3.jpg")

im4=pygame.image.load("C:/Users/admin/OneDrive/Pictures/Screenshots/4.jpg")

im5=pygame.image.load("C:/Users/admin/OneDrive/Pictures/Screenshots/5.jpg")

im6=pygame.image.load("C:/Users/admin/OneDrive/Pictures/Screenshots/6.jpg")

im7=pygame.image.load("C:/Users/admin/OneDrive/Pictures/Screenshots/7.jpg")

im8=pygame.image.load("C:/Users/admin/OneDrive/Pictures/Screenshots/8.jpg")

im9=pygame.image.load("C:/Users/admin/OneDrive/Pictures/Screenshots/9.jpg")

im1=pygame.transform.scale(im1,(175,175))

im2=pygame.transform.scale(im2,(175,175))

im3=pygame.transform.scale(im3,(175,175))

im4=pygame.transform.scale(im4,(175,175))

im5=pygame.transform.scale(im5,(175,175))

im6=pygame.transform.scale(im6,(175,175))

im7=pygame.transform.scale(im7,(175,175))

im8=pygame.transform.scale(im8,(175,175))

im9=pygame.transform.scale(im9,(175,175))

#loading and resizing image to display GAME OVER in 2048 game

Goim=pygame.image.load("C:/Users/admin/OneDrive/Pictures/Screenshots/game over.jpg")

Goim=pygame.transform.scale(Goim,(400,200))

#connecting to mysql and obtaining all data stored

import mysql.connector as m

mo=m.connect(host='localhost',user='root',password='mineymice',database='user')

co=mo.cursor()

'''co.execute('use user')

co.execute('create table user\_data(p\_no integer primary key,username char(40) not null,points integer,password char(40) not null)')'''

k=True

# loop to allow different users to use one after other without closing whole application

while k:

co.execute('select \* from user\_data')

data=co.fetchall()

rec=[] #list containing data of user

tk=1 #variable used to record if quit pressed in password or username window and then not go through any other process

#displaying page for entering username

a=True

s=''

while a:

pygame.display.set\_caption('Username') #caption/title

event=pygame.event.poll()

keys=pygame.key.get\_pressed()

if event.type==pygame.QUIT:

k=False

a=False

tk=0

if event.type==pygame.KEYDOWN:

key=pygame.key.name(event.key)

if len(key)==1:

if keys[pygame.K\_LSHIFT] or keys[pygame.K\_RSHIFT]:

s+=key.upper()

else:

s+=key

elif key=='backspace':

s=s[:len(s)-1]

elif event.key==pygame.K\_RETURN:

a=False

st='Enter user\_name (of player 1 for X-O game) in black rectangle'

sp="Don't use Space,Underscore."

sm='Press Enter to move to page to enter password'

tp=fo.render(sp,1,(225,225,225))

tm=fo.render(sm,1,(225,225,225))

tx=fo.render(st,1,(225,225,225))

tn=fo.render(s,1,(225,225,225))

dis.fill((26,159,174))

pygame.draw.rect(dis,(0,0,0),pygame.Rect(45,290,500,45))

dis.blit(tx,(25,50))

dis.blit(tn,(50,300))

dis.blit(tp,(25,125))

dis.blit(tm,(25,200))

pygame.display.update()

re=0 #variable used to record if username already exists or not

#recording data of user in list rec

for i in data:

if i[1]==s:

for j in i:

rec.append(j)

re+=1

# displaying page for entering password

b=True

u=''

while b and tk:

pygame.display.set\_caption('Password') #caption/title

event=pygame.event.poll()

keys=pygame.key.get\_pressed()

if event.type==pygame.QUIT:

b=k=False

tk=0

if event.type==pygame.KEYDOWN:

key=pygame.key.name(event.key)

if len(key)==1:

if keys[pygame.K\_LSHIFT] or keys[pygame.K\_RSHIFT]:

u+=key.upper()

else:

u+=key

elif key=='backspace':

u=u[:len(u)-1]

elif event.key==pygame.K\_RETURN:

if re!=0:

if u==rec[3]:

b=False

else:

u=''

else:

b=False

st='Enter password (of player 1 for X-O game) in black rectangle'

sp="No Space, Underscore. If page appears again: enter different"

sp1='password as that is not the correct one.'

sm='Press Enter to move to go to home page'

tp1=fo.render(sp1,1,(225,225,225))

tm=fo.render(sm,1,(225,225,225))

tx=fo.render(st,1,(225,225,225))

tu=fo.render(u,1,(225,225,225))

tp=fo.render(sp,1,(225,225,225))

dis.fill((26,159,174))

pygame.draw.rect(dis,(0,0,0),pygame.Rect(45,290,500,45))

dis.blit(tx,(25,50))

dis.blit(tu,(50,300))

dis.blit(tp,(25,105))

dis.blit(tp1,(25,125))

dis.blit(tm,(25,180))

pygame.display.update()

#adding record of user if it doesn't exist

if re==0 and tk==1:

co.execute('select max(p\_no) from user\_data')

lpno=co.fetchone()

mpno=lpno[0]

ins='insert into user\_data values(%s,%s,%s,%s)'

if mpno==None:

rectu=(1,s,0,u)

co.execute(ins,rectu)

rec=[1,s,0,u]

else:

rectu=(mpno+1,s,0,u)

co.execute(ins,rectu)

rec=[mpno+1,s,0,u]

mo.commit()

if tk==1:

p=rec[2] #points of user

#loop for home page, includes both games, user profile and leaderboard

z=True

while z and tk:

dis=pygame.display.set\_mode((600,600)) #display

pygame.display.set\_caption('Home Page') #caption/title

for event in pygame.event.get():

if event.type==pygame.QUIT:

z=False

k=False

elif event.type==pygame.KEYDOWN:

#to enter different username

if event.key==pygame.K\_4:

z=False

# playing tic tac toe if 1 pressed

elif event.key==pygame.K\_1:

pygame.display.set\_caption('Tic Tac Toe Game') #caption/title

l=[[1,2,3],[4,5,6],[7,8,9],[1,4,7],[2,5,8],[3,6,9],[1,5,9],[3,5,7]] #list of all possible combination to win

a=[] #list to record all boxes already used

xl=[] #list to record boxes used by player 1 - X

ol=[] #list to record boxes used by player 2 - O

c=1 #integer value of the number of the next turn

e=1 #integer value of the total number of turns already done

n=0 #number of box chosen by the player

r=True

d=0

zk=1 #variable used to not display conclusion pages or add points if quit pressed in middle of game

while c<=9:

#coloring the background and drawing the 4 lines that will split the display into 9 boxes

dis.fill((225,225,225))

pygame.draw.line(dis,(0,0,0),(200,0),(200,600),5)

pygame.draw.line(dis,(0,0,0),(400,0),(400,600),5)

pygame.draw.line(dis,(0,0,0),(0,200),(600,200),5)

pygame.draw.line(dis,(0,0,0),(0,400),(600,400),5)

#bliting images of the numbers of the squares

dis.blit(im1,(10,10))

dis.blit(im2,(210,10))

dis.blit(im3,(410,10))

dis.blit(im4,(10,210))

dis.blit(im5,(210,210))

dis.blit(im6,(410,210))

dis.blit(im7,(10,410))

dis.blit(im8,(210,410))

dis.blit(im9,(410,410))

pygame.display.update()

while r:

for event in pygame.event.get():

#closing game if user quits

if event.type==pygame.QUIT:

r=False

z=False

k=False

c=10

zk=0

# recording number of box chosen by the player and the cordinates where the image of X/O will be blitted

elif event.type==pygame.KEYDOWN:

if event.key==pygame.K\_1:

n,x,y=1,10,10

elif event.key==pygame.K\_2:

n,x,y=2,210,10

elif event.key==pygame.K\_3:

n,x,y=3,410,10

elif event.key==pygame.K\_4:

n,x,y=4,10,210

elif event.key==pygame.K\_5:

n,x,y=5,210,210

elif event.key==pygame.K\_6:

n,x,y=6,410,210

elif event.key==pygame.K\_7:

n,x,y=7,10,410

elif event.key==pygame.K\_8:

n,x,y=8,210,410

elif event.key==pygame.K\_9:

n,x,y=9,410,410

#bliting the image of X/O at required positions

if c%2!=0:

if n not in a:

if n!=0:

dis.blit(Xim,(x,y))

pygame.display.update()

xl.append(n)

a.append(n)

e=c

c+=1

else:

if n not in a:

if n!=0:

dis.blit(Oim,(x,y))

pygame.display.update()

ol.append(n)

a.append(n)

e=c

c+=1

#checking if either player won by comaring xl/ol with l and closing game if they did

for i in l:

if e%2!=0 and zk:

if i[0] in xl:

if i[1] in xl:

if i[2] in xl:

r=False

xl=[]

d=1

p+=10

else:

if i[0] in ol and zk:

if i[1] in ol:

if i[2] in ol:

r=False

ol=[]

d=1

t.tick(60)

#closing game if all 9 turns are done and nobody has won yet

if e==9 and zk:

if d==0:

r=False

print("It's a draw, nobody lost !!!")

p+=5

#displaying winner of game of draw

while c<=10 and zk:

for event in pygame.event.get():

if event.type==pygame.QUIT:

c=11

r=k=z=False

if event.type==pygame.KEYDOWN:

if event.key==pygame.K\_RETURN:

c=11 #to come out of while c<=9 and c<=10 loop

# displaying who won or draw

if d==1:

if e%2!=0:

dis.fill((0,0,0))

dis.blit(Pxim,(50,100))

pygame.display.update()

else:

dis.fill((0,0,0))

dis.blit(Poim,(50,100))

pygame.display.update()

elif d==0:

dis.fill((0,0,0))

dis.blit(Pdim,(50,100))

pygame.display.update()

t.tick(60)

# displaying leaderboard

elif event.key==pygame.K\_3:

g=True

while g:

pygame.display.set\_caption('Leaderboard') #caption/title

cr=70

for event in pygame.event.get():

if event.type==pygame.QUIT:

g=z=k=False

if event.type==pygame.KEYDOWN:

if event.key==pygame.K\_RETURN:

g=False

dis.fill((26,159,174))

co.execute('select \* from user\_data order by points desc,p\_no')

ld=co.fetchall()

sl='LEADERBOARD (Top 10)'

sr='Rank Username- Points'

se='Press Enter to return to home page'

tx=fo.render(se,1,(225,225,225))

tr=fo.render(sr,1,(225,225,225))

tl=fo.render(sl,1,(225,225,225))

dis.blit(tl,(50,15))

dis.blit(tr,(50,75))

dis.blit(tx,(50,550))

q=1

for i in ld[:10]:

cr+=45

sld=str(q)+'.'+' '+i[1]+'- '+str(i[2])

tld=fo.render(sld,1,(225,225,225))

dis.blit(tld,(50,cr))

q+=1

pygame.display.update()

# displaying user profile

elif event.key==pygame.K\_SPACE:

y=True

while y:

pygame.display.set\_caption('User Profile') #caption/title

for event in pygame.event.get():

if event.type==pygame.QUIT:

y=z=k=False

if event.type==pygame.KEYDOWN:

if event.key==pygame.K\_RETURN:

y=False

# to delete user record

elif event.key==pygame.K\_1:

tpl=(rec[0],)

ex='delete from user\_data where p\_no=%s'

co.execute(ex,tpl)

mo.commit()

y=False

z=False

su='Password : '+u

sn='Username : '+s

spno='Player number : '+str(rec[0])

sp='Points : '+str(p)

se='Press Enter to return to home page'

sd='Press 1 to delete your profile'

tx=fo.render(se,1,(225,225,225))

tp=fo.render(sp,1,(225,225,225))

tpno=fo.render(spno,1,(225,225,225))

td=fo.render(sd,1,(225,225,225))

tsu=fo.render(su,1,(225,225,225))

tsn=fo.render(sn,1,(225,225,225))

dis.fill((26,159,174))

dis.blit(tsu,(50,250))

dis.blit(tx,(50,500))

dis.blit(tsn,(50,150))

dis.blit(tp,(50,350))

dis.blit(tpno,(50,50))

dis.blit(td,(50,550))

pygame.display.update()

#playing 2048 game if 2 is pressed

elif event.key==pygame.K\_2:

rk=1 #used so that game over page not displayed if quit pressed in middle of game

v=True

while v:

ste='Press Enter to return to home screen'

tse=fo.render(ste,1,(0,0,0))

m = Main()

# TO DO

# Refactor move() to loop functions

h=True

while h and rk:

for event in pygame.event.get():

if event.type==pygame.QUIT:

h=z=k=False

if event.type==pygame.KEYDOWN:

if event.key==pygame.K\_RETURN:

h=False

dis.fill((0,0,0))

dis.blit(Goim,(50,100))

pygame.display.update()

#updating points of user

rec[2]=p

tup=(rec[2],rec[0])

exc='update user\_data set points=%s where p\_no=%s'

co.execute(exc,tup)

mo.commit()

# home page display

dis.fill((0,0,0))

dis.blit(TTTim,(25,135))

dis.blit(Upim,(312,25))

dis.blit(Lbim,(25,25))

dis.blit(Ptim,(25,445))

dis.blit(Luim,(386,135))

pygame.display.update()

t.tick(60)

co.close()

mo.close()

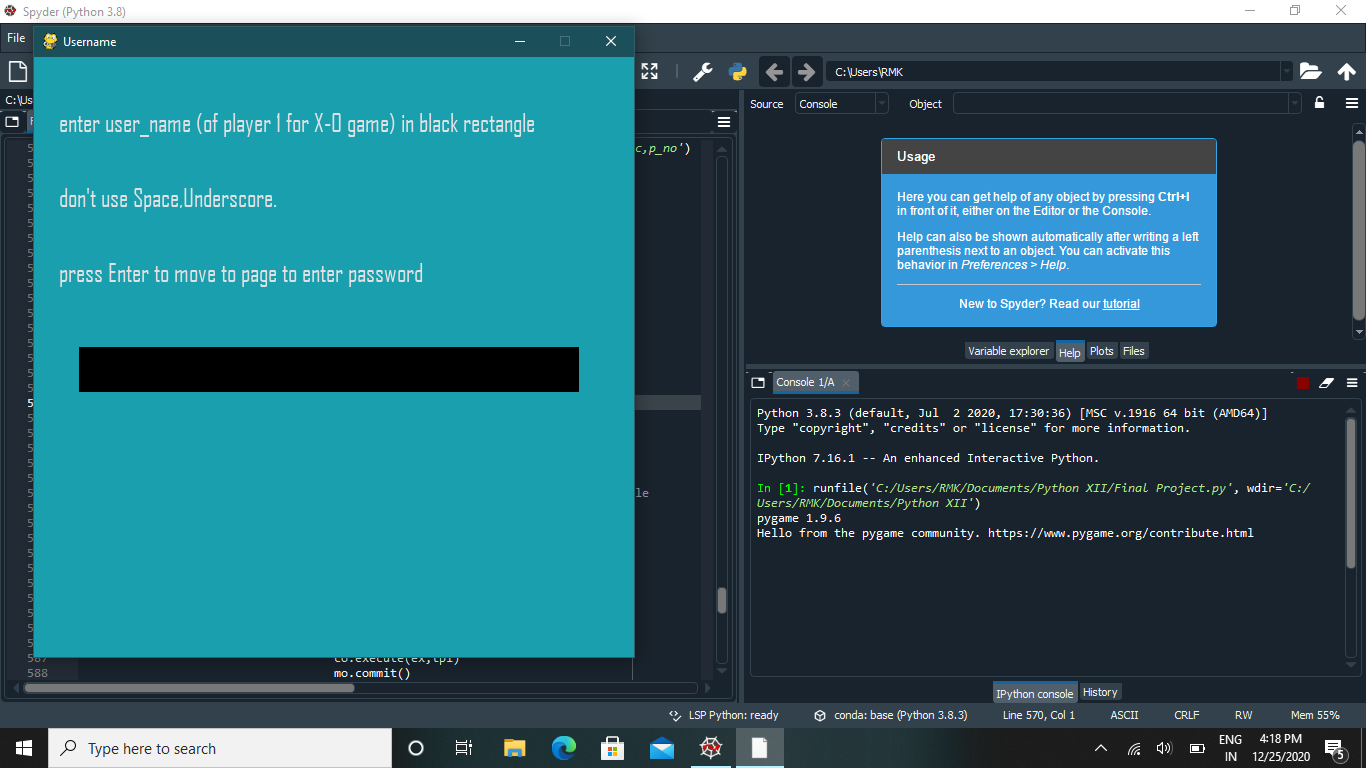
pygame.quit()

**OUTPUT**

The Username input screen:

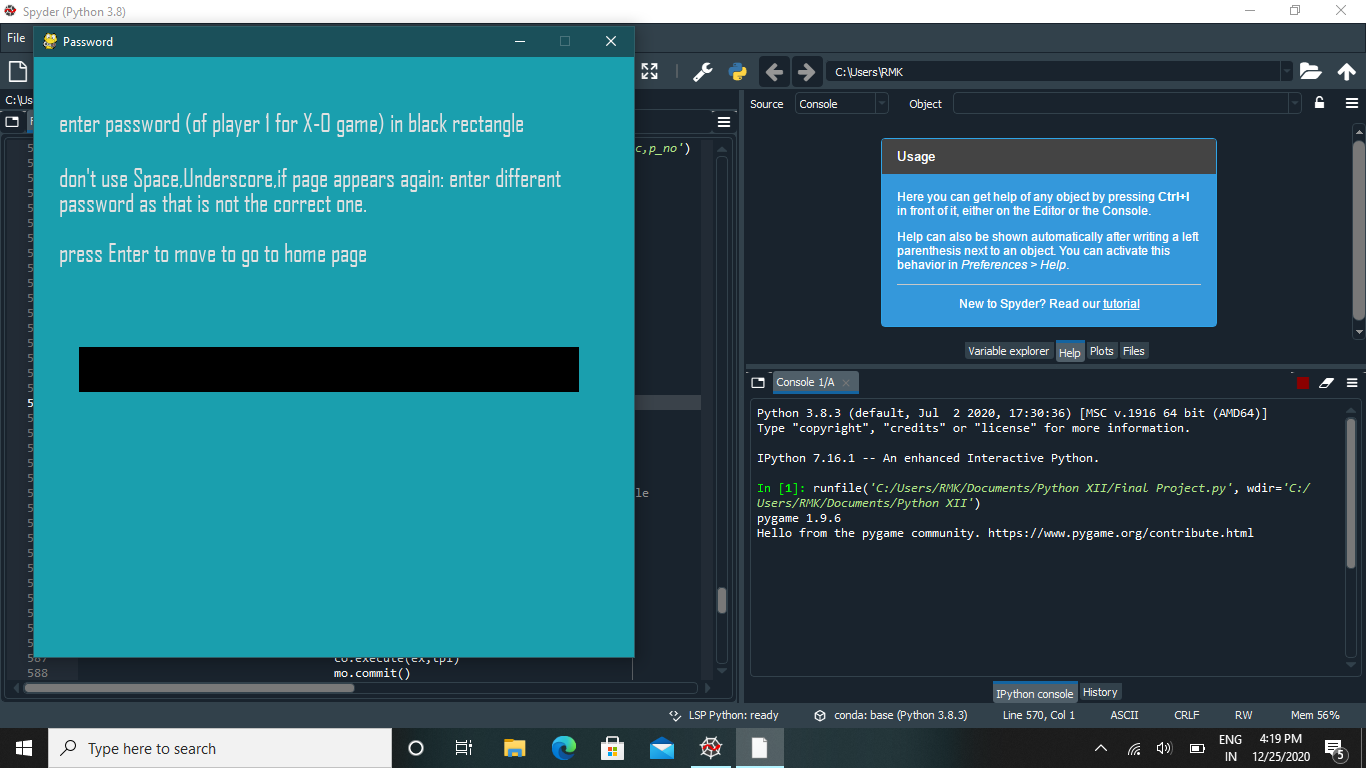
This page shows up first when the application is run, the username has to be typed in the black box.

If the entered username exists, then the already existing record is accessed otherwise a new record will be created once the user inputs the password with the points as zero and the player number as the number 1 more than the maximum player number that already exists.



The Password input screen:

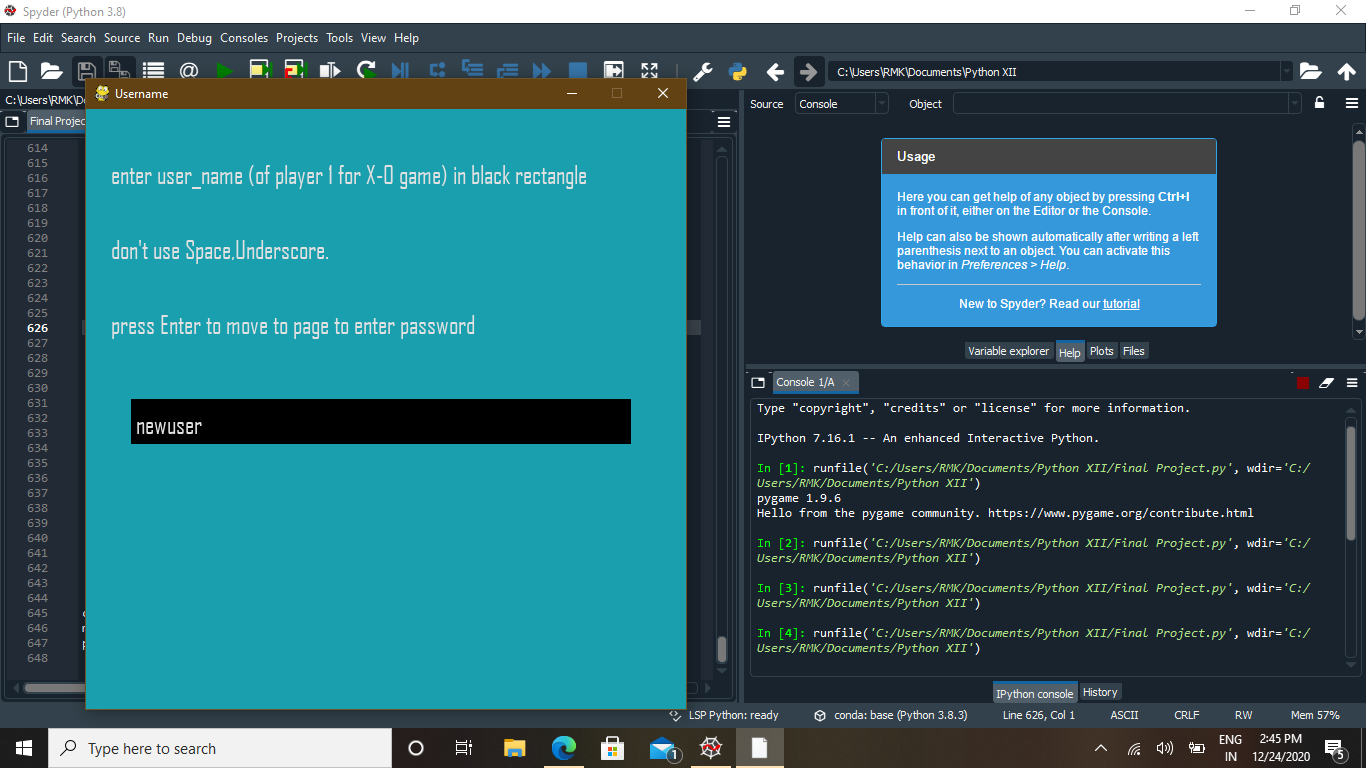
If the entered username exists, then the password entered here has to be the same as the one saved in the record which has that username otherwise if we enter anything else and press enter, the black will just become empty again prompting the user to enter another password. If the username doesn’t already exist then whatever is typed in as password will be saved as the password corresponding to the new username in the new record that will be created.



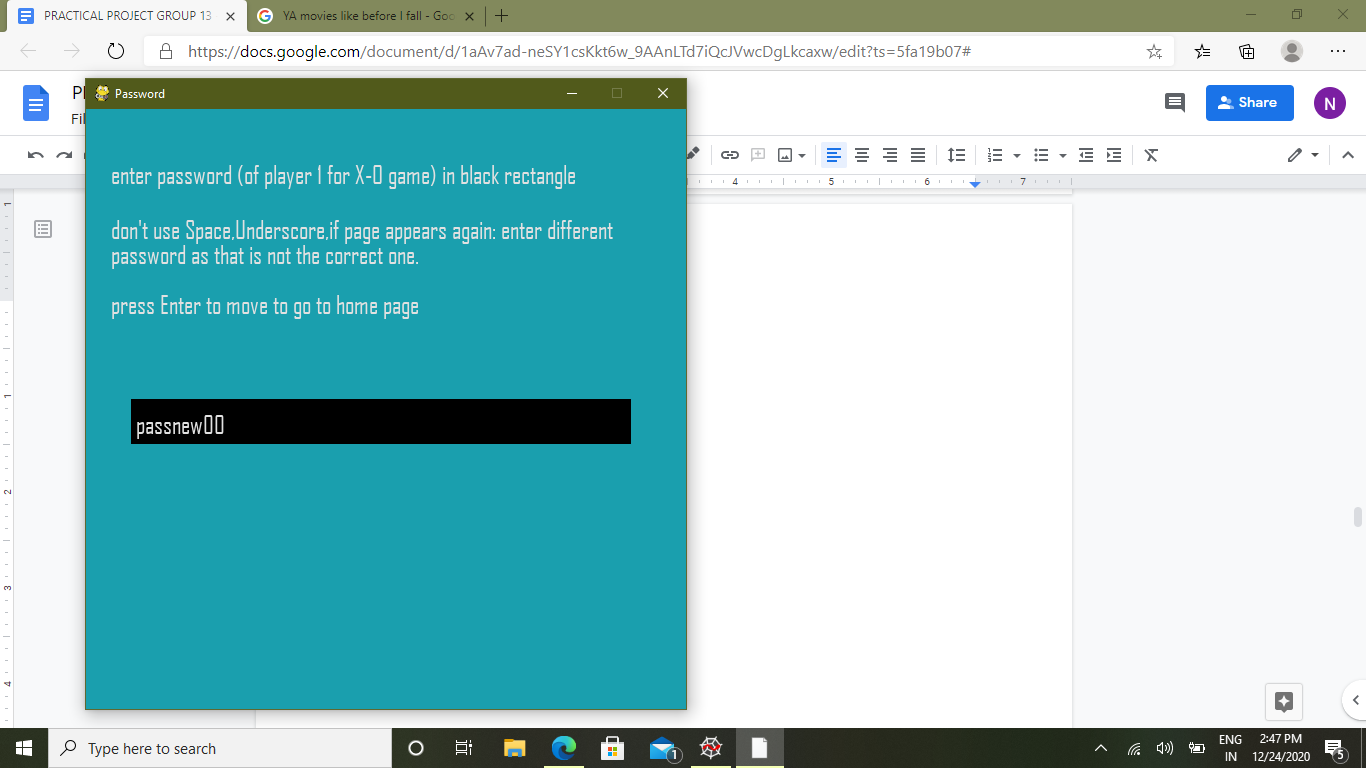
How the user profiles will be accessed if it’s a new user or an old user who logs in :

If new user is using the program :

Username input :

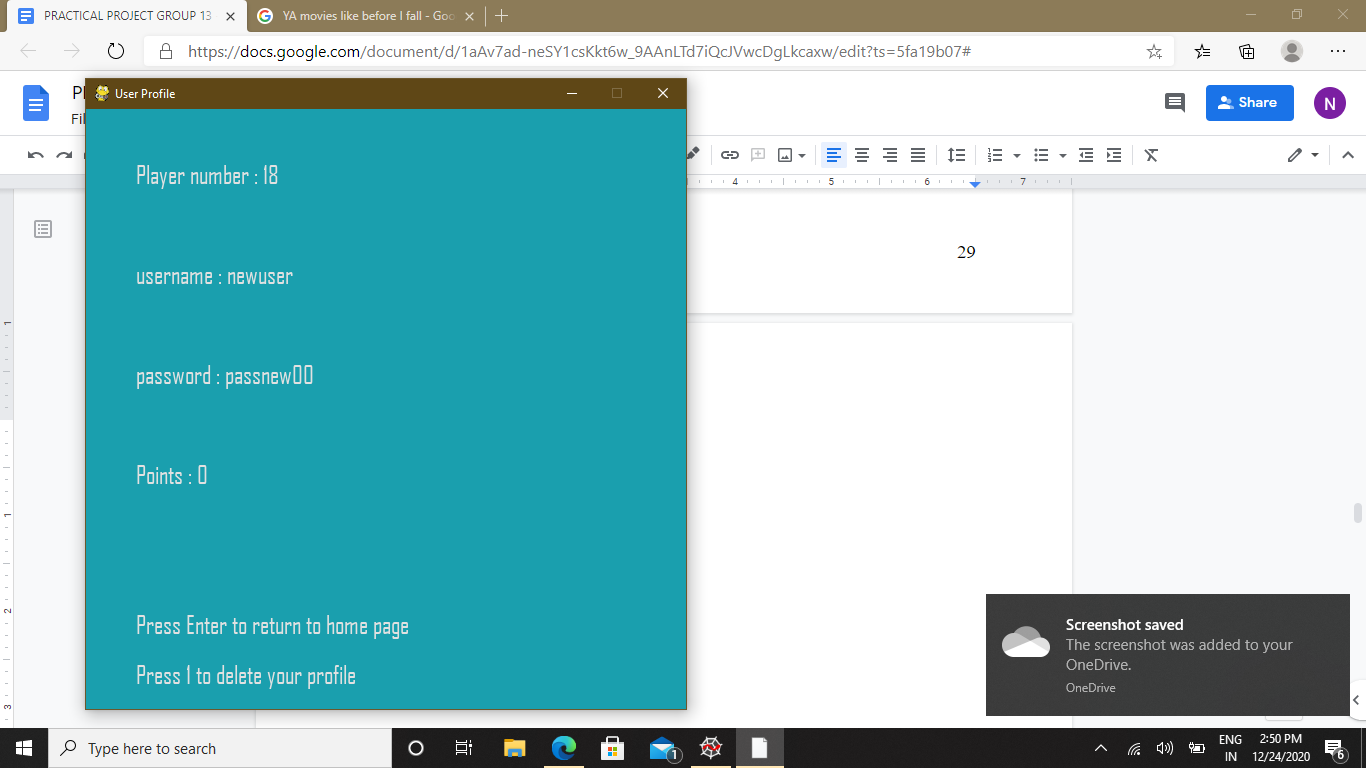


Password input :



User Profile :

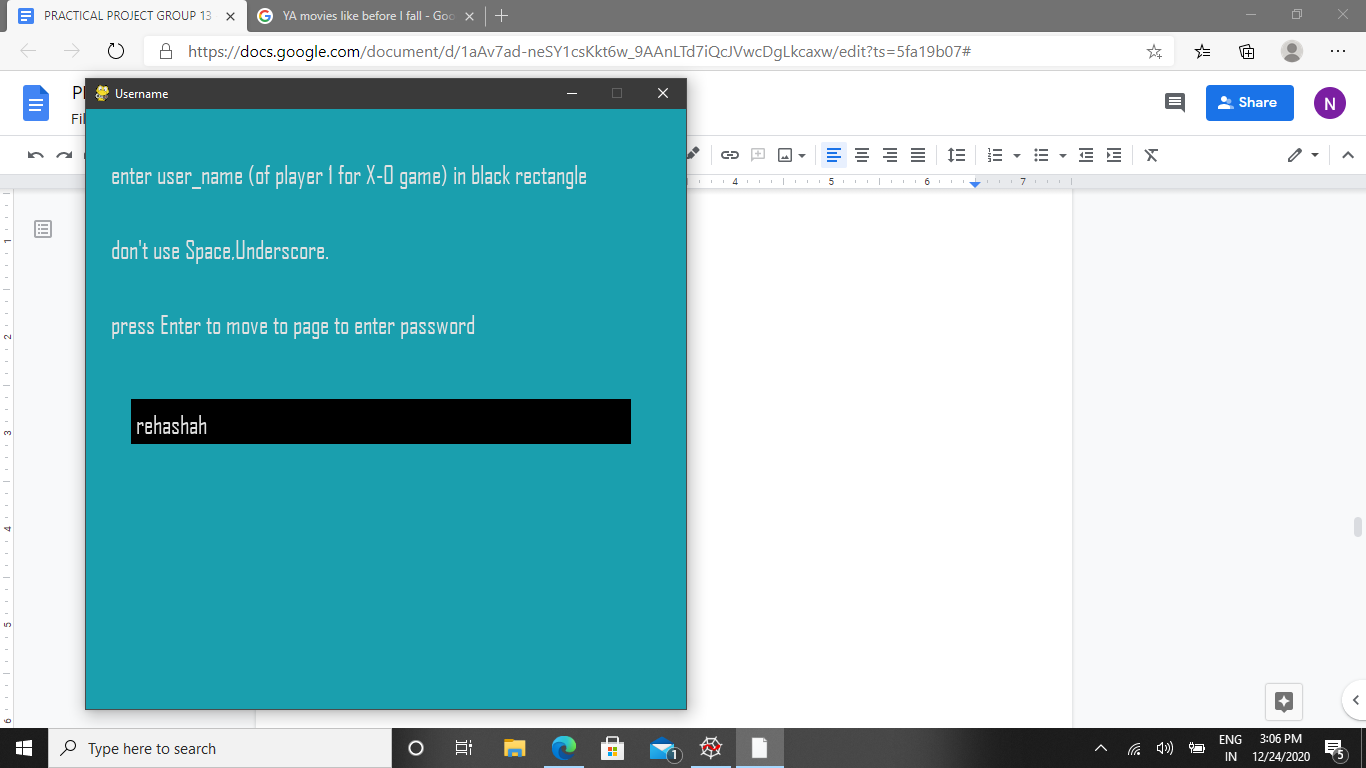
This page was accessed after pressing enter on the password input page, going to the home page and then pressing space on the home page.



Here, we can see that the points of the user are zero as it’s a new account. Also, the player number of the new user is 18, indicating that the highest player number that existed until now is 17.

If old user is using the program :

Username input :

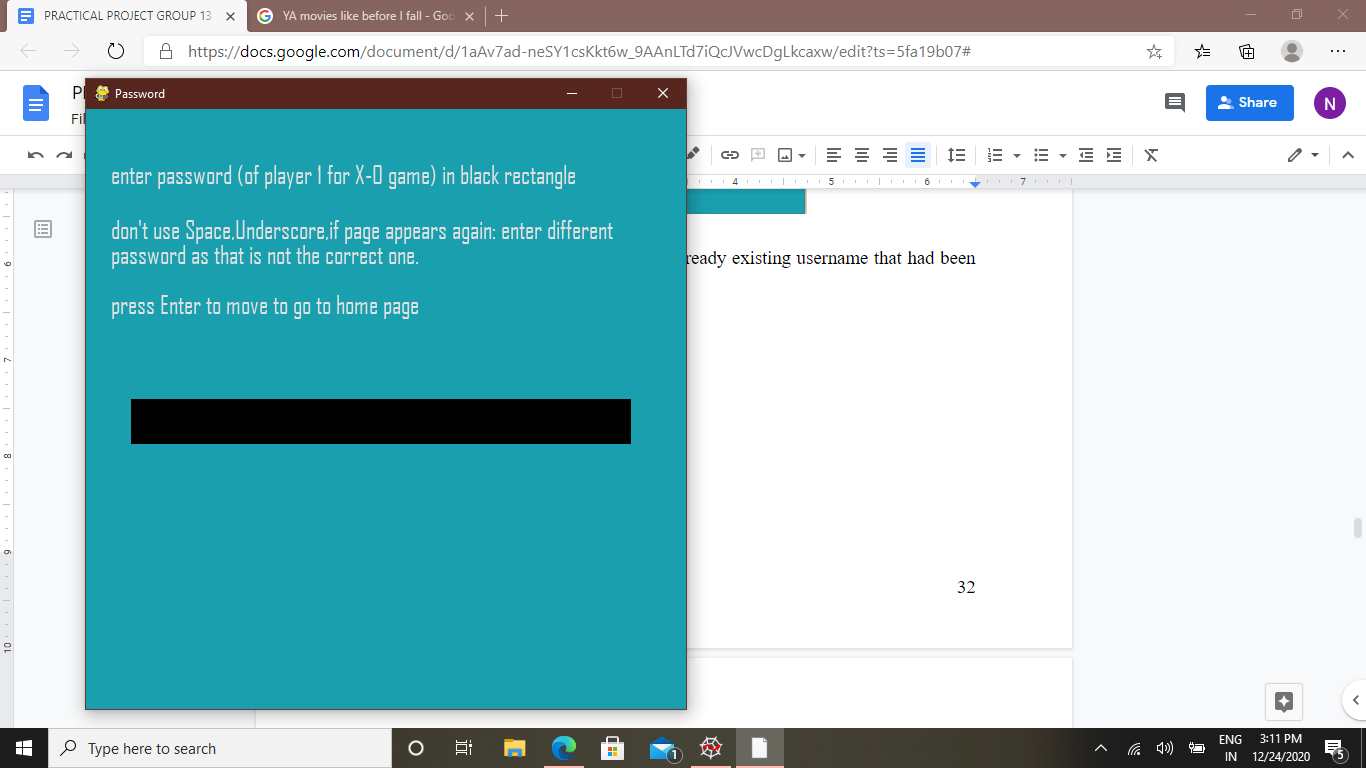


Password input :

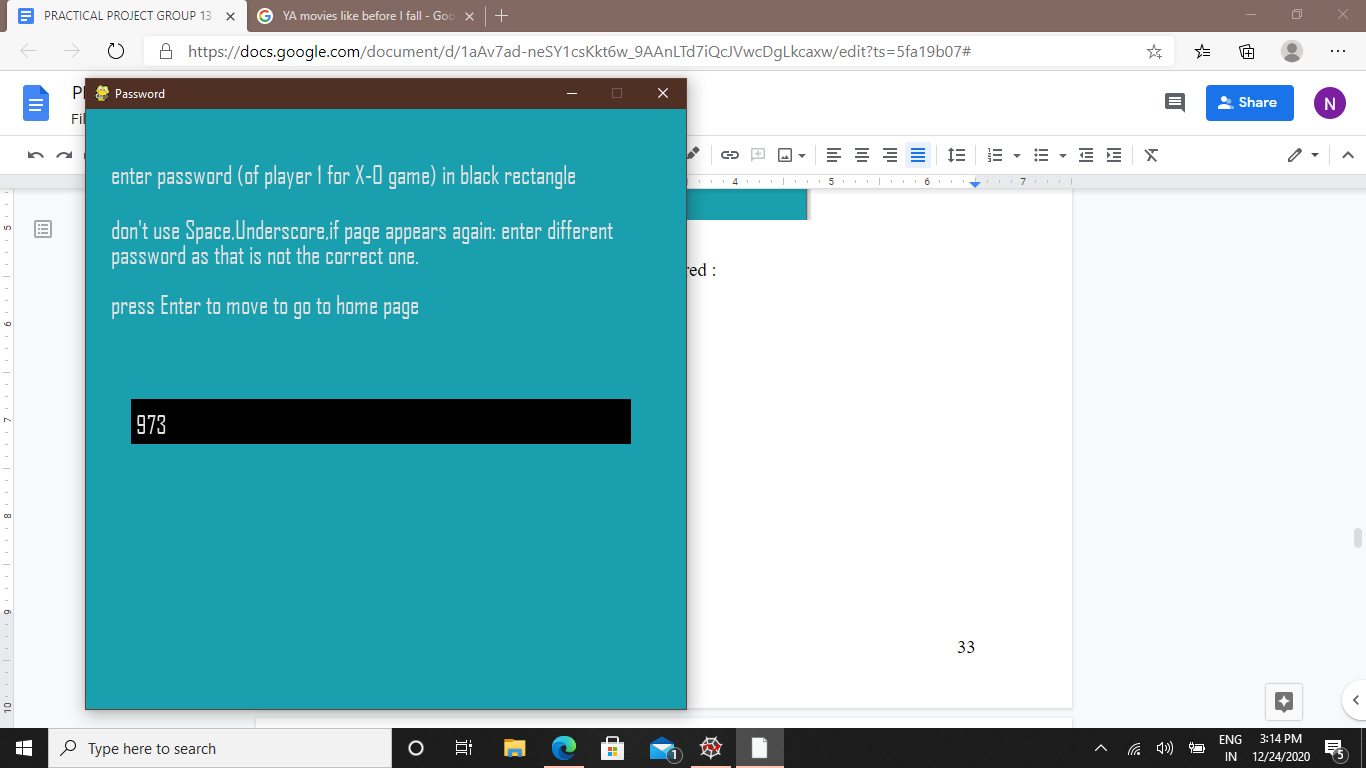


Here, this is the wrong password for the already existing username that had been entered.

As the password entered was incorrect, the black box will become empty when enter is pressed with wrong password in black box and this will come again :



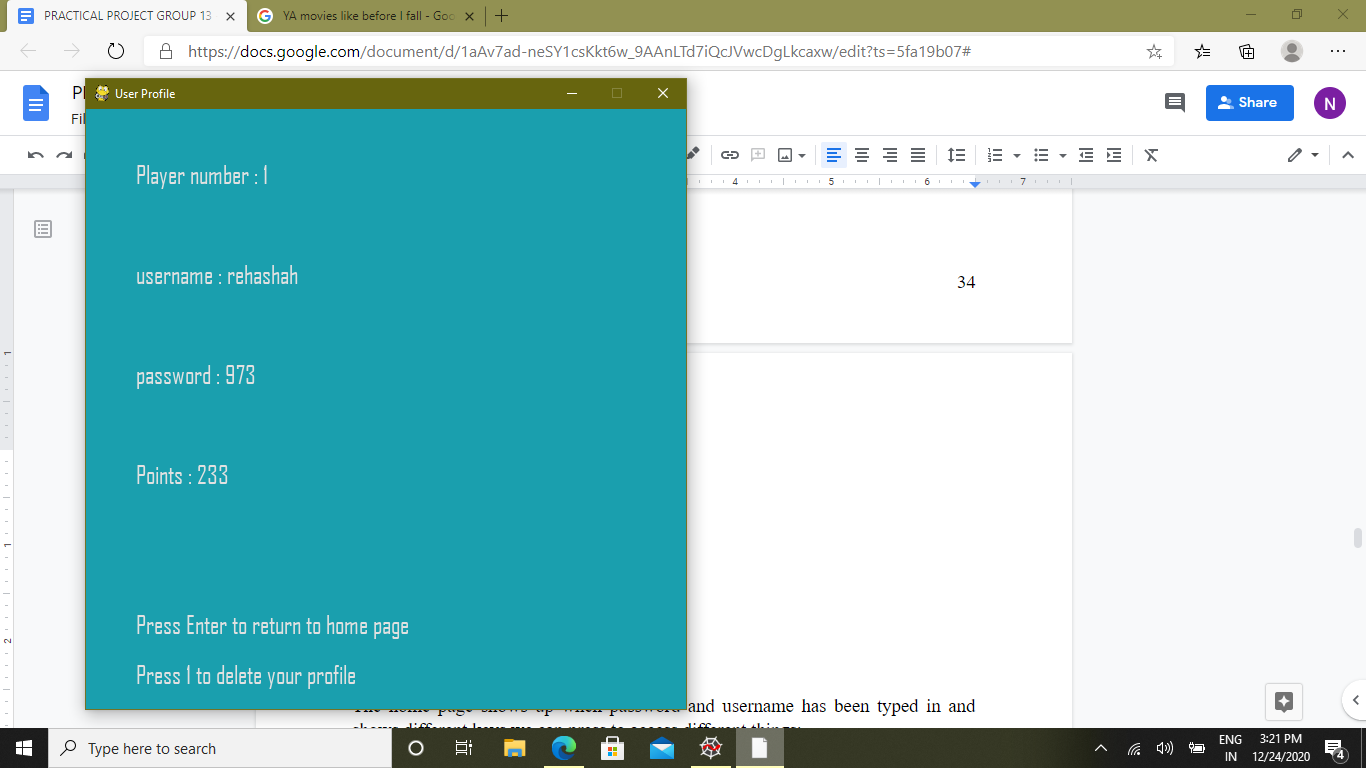
And then, when the correct password is entered :



Here, as the password entered is correct, on pressing enter the home screen will appear.

User Profile :

This page can be accessed by pressing space on the home page.



Here, we can see that the points the user has earned till now have been added and shown here, in this case, 233.

The Home Page :

The home page shows up when password and username has been typed in and shows different keys we can press to access different things:

Leaderboard - 3

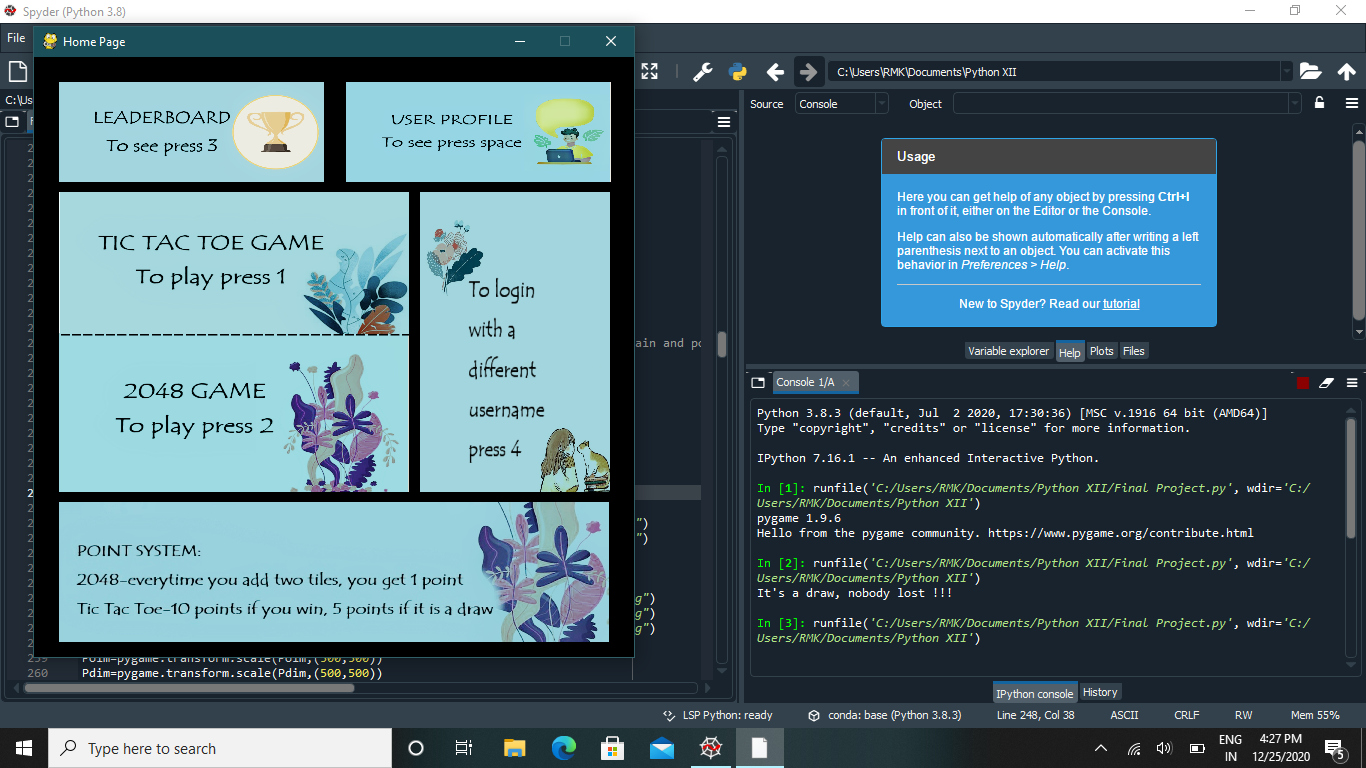
User Profile - space

Tic Tac Toe game - 1

2048 game - 2

Login with different account - 4

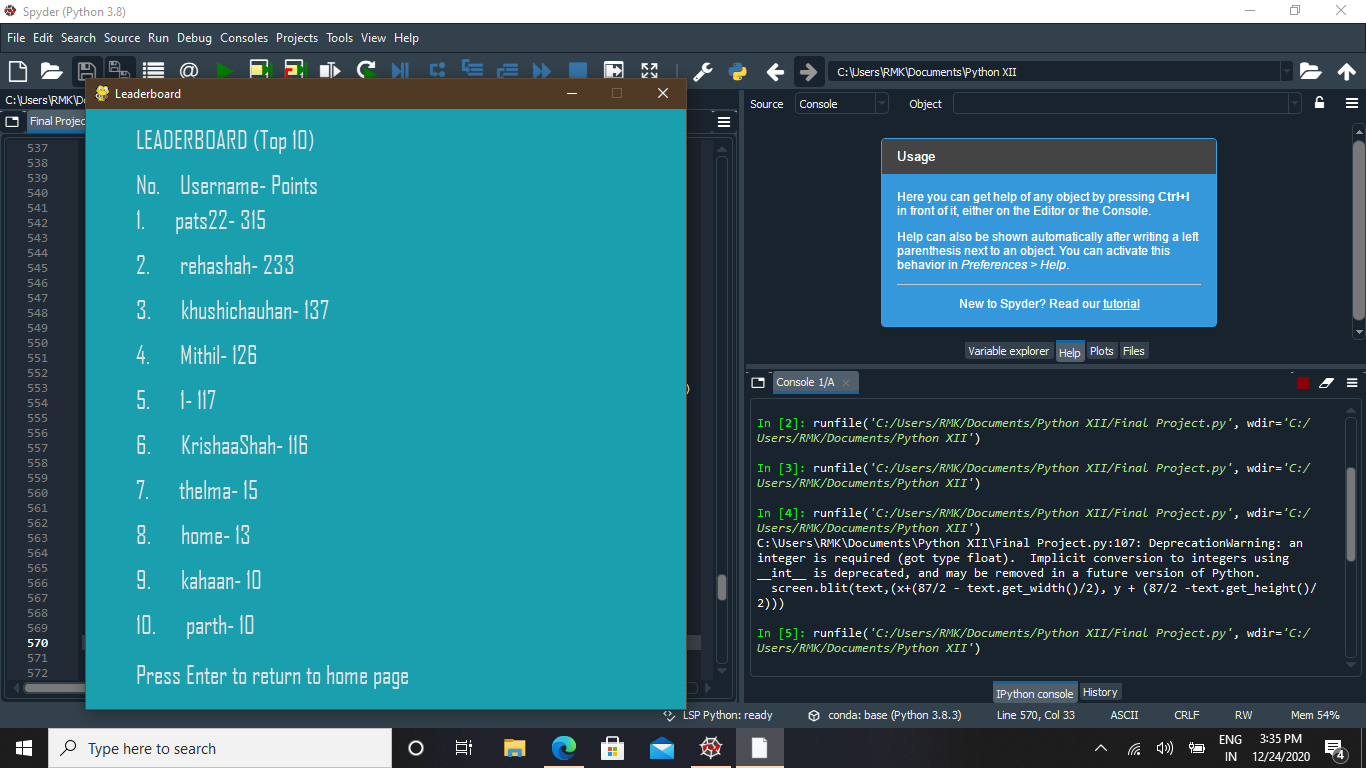
It also gives a description of the point system used to add the points of the user.



The Leaderboard Page:

This page is shown when the user presses 3 on the home page.Here, the program accesses records sorted by points from the SQL table and displays the top ten. To return to the home page, the user has to press Enter.

If the points of 2 users are the same, then the user who created an account first will be displayed first.



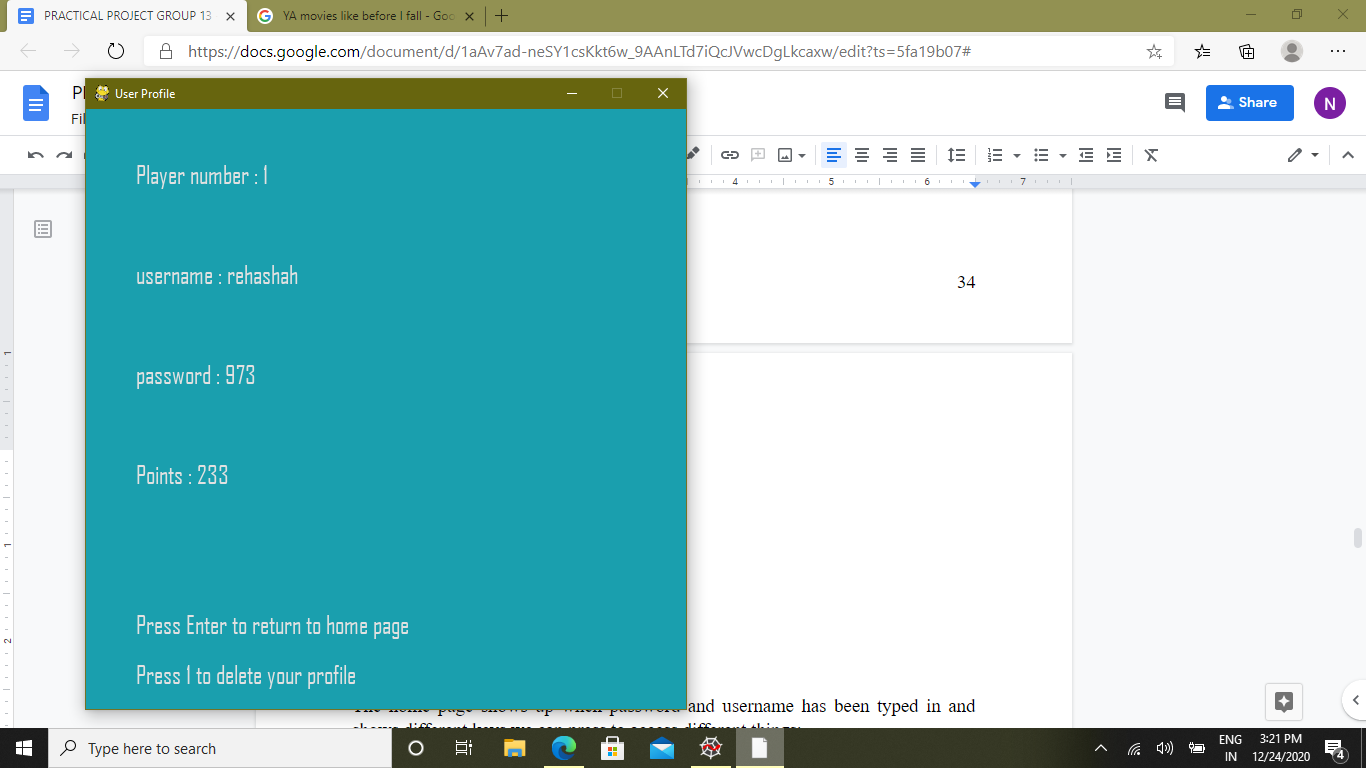
The User Profile Page:

This page can be viewed by pressing space on the home page. Here, the data ( the player number, username,password and points ) in the particular user’s record is displayed. The points displayed will change every time the user plays a game and earns some.

If the user wants to delete their account then they can press 1 to do so. If the user presses 1 then the current record is deleted and the username page appears again. Here they can enter another username and password and enter the application again or press the red cross button ( quit button ) and exit the whole application.

The player numbers of the deleted records will not be used again so in a list of player numbers, there will be a gap in these places. For example : if there are 5 players with player numbers 1, 2, 3, 4, 5 and player 3 deletes their account, then the player numbers will be 1, 2, 4, 5 and the next new player will get the number 6( which 1 plus to the maximum number - 5 ).

To go back to the home page, the user has to press Enter.

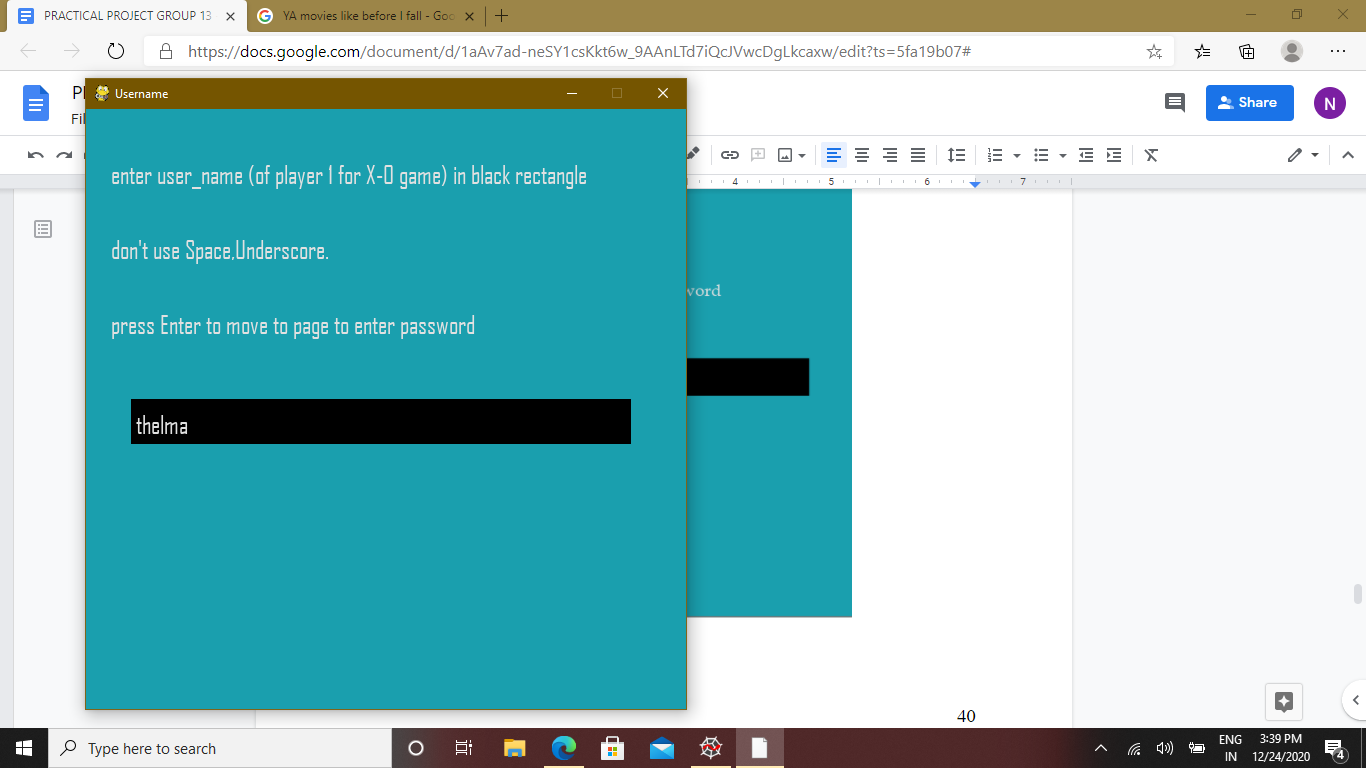


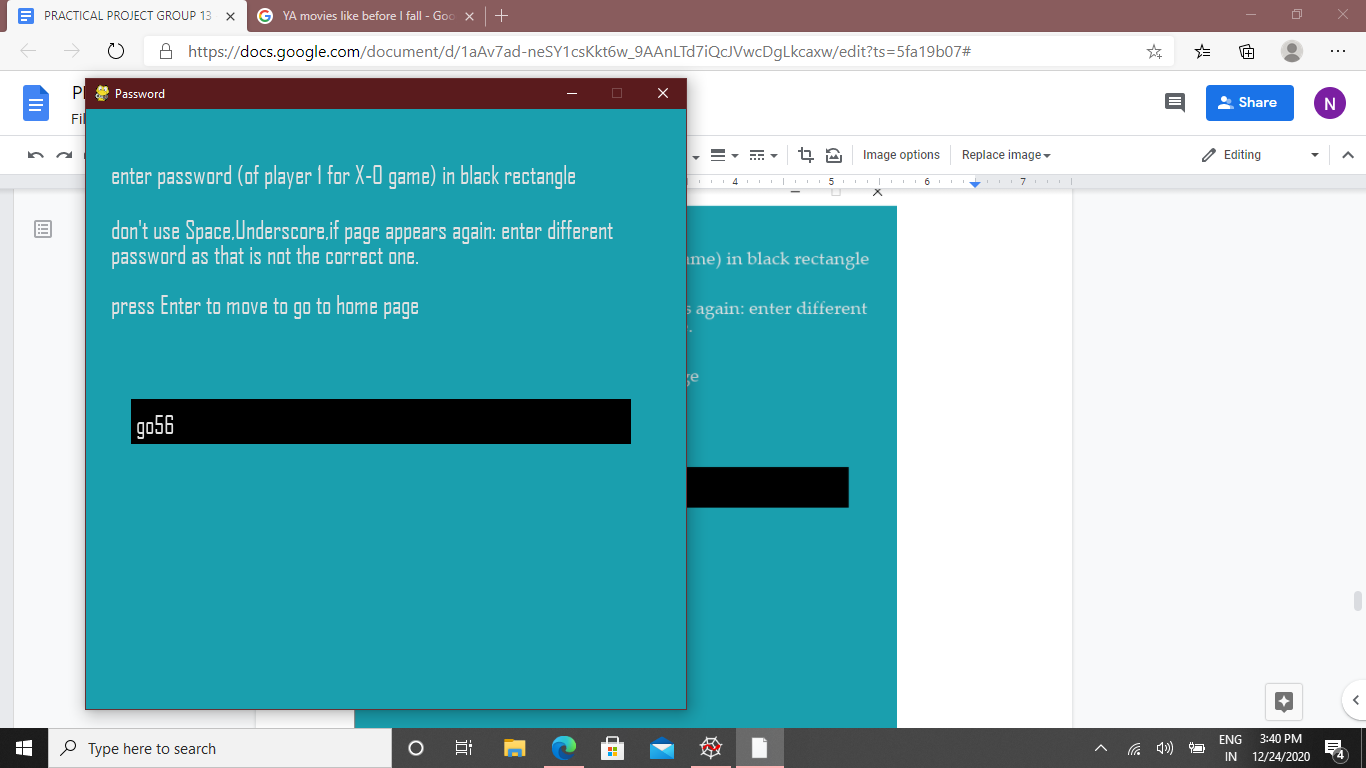
To login with different account :

If the user presses 4 when they are on the home page, it will take them back to the username page and then the password page where the user or a different user can login with another username and password. After entering the password, the homepage will be displayed and the whole application can be used again with

another account without closing the application completely.

Here, a new or an old user can login with the procedure being the same as when they login at the beginning of the program.



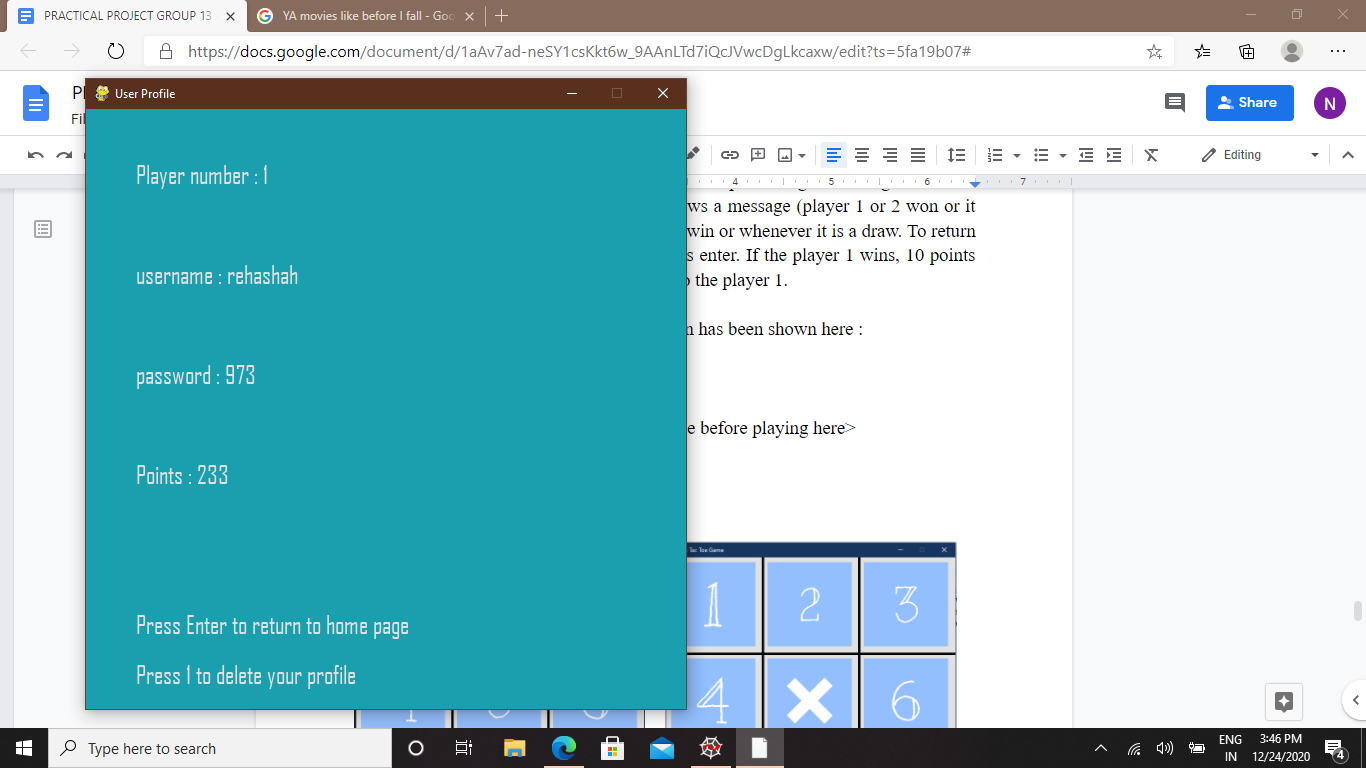


The Tic Tac Toe Game:

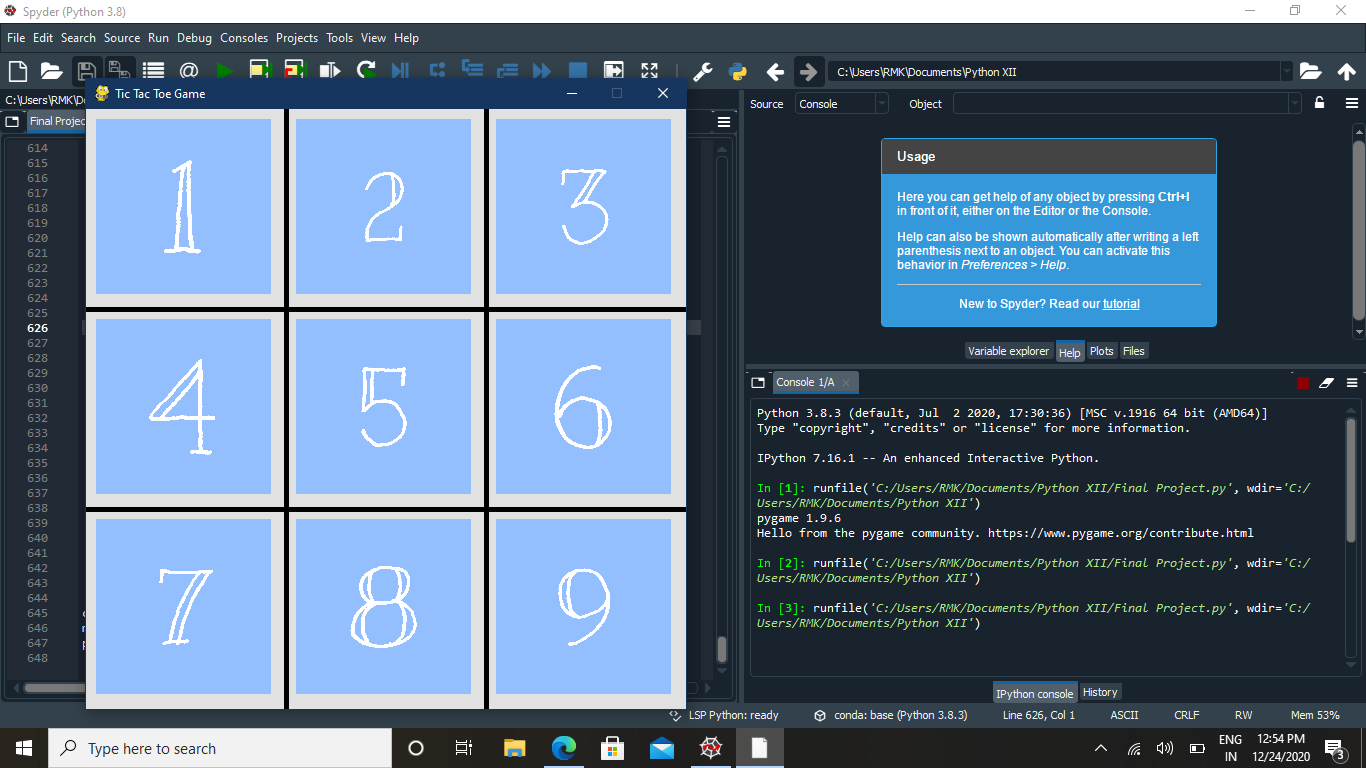
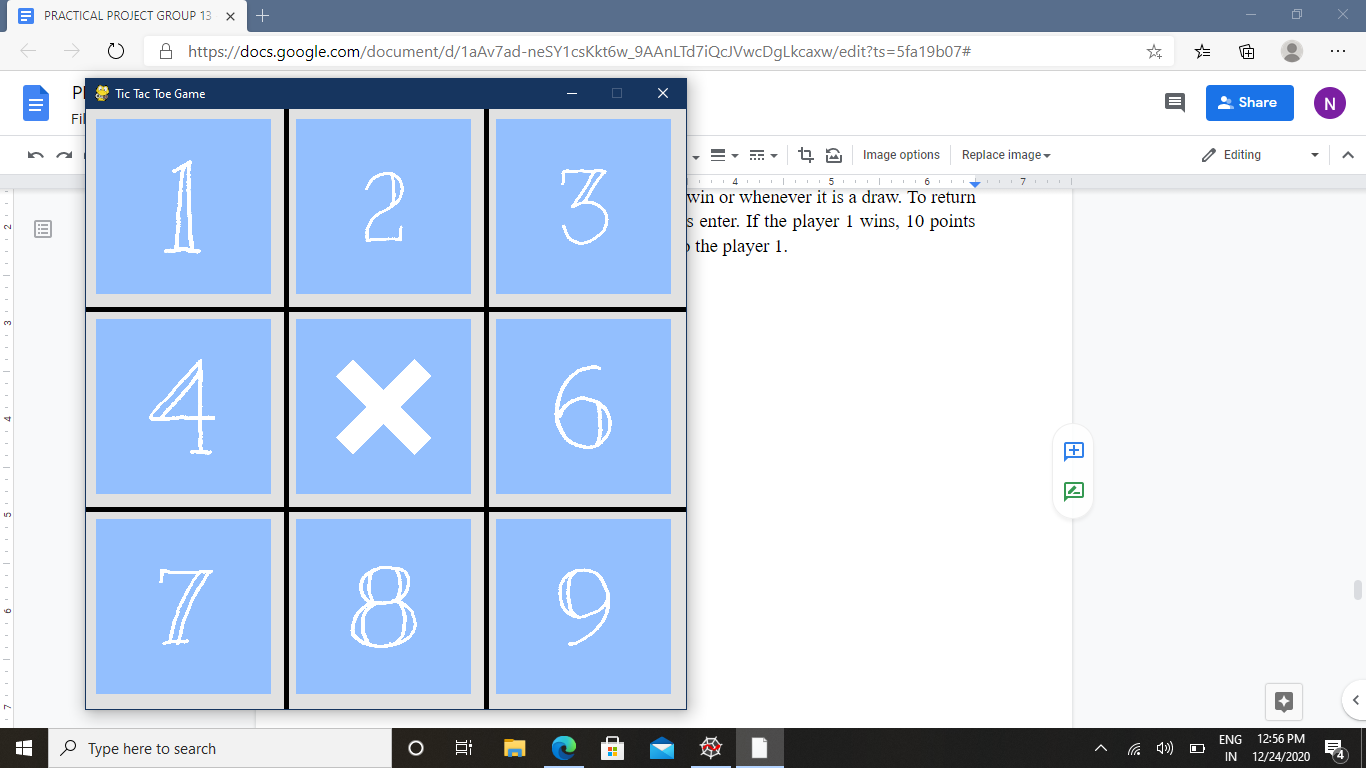
This is the classic game of X’s and O’s. User presses 1, when on the home page, to play. Here, the game works by pressing numbers on the keyboard. The user whose account has been used to login is the player 1 representing “X” and gets the first turn. The game automatically stops and shows a message (player 1 or 2 won or it was a draw ) whenever either of the players win or whenever it is a draw. To return to the homescreen, the player needs to press enter. If the player 1 wins, 10 points are given. If it is a draw, 5 points are given to the player 1.

A game in which the user (Player 1 “X”) won has been shown here :

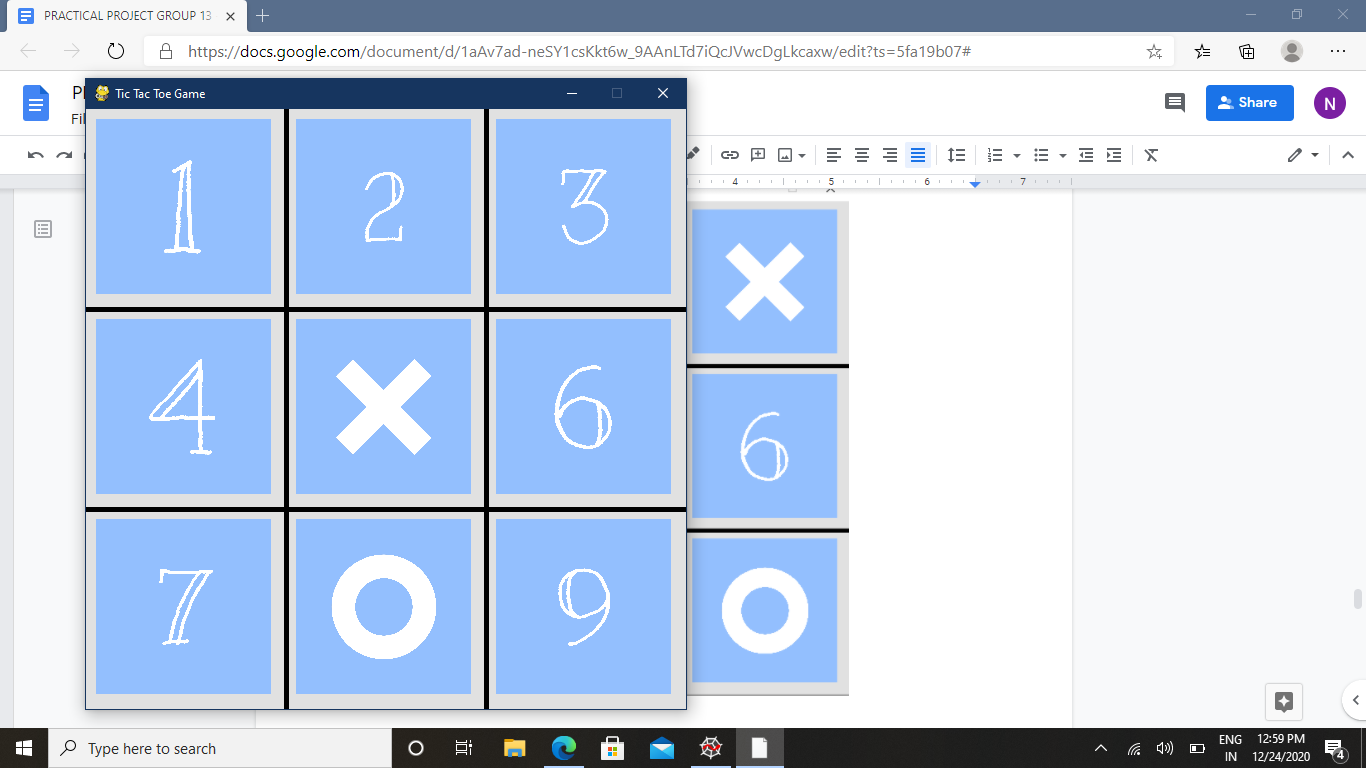
Points of user before the game : ( can be seen here as 233 )



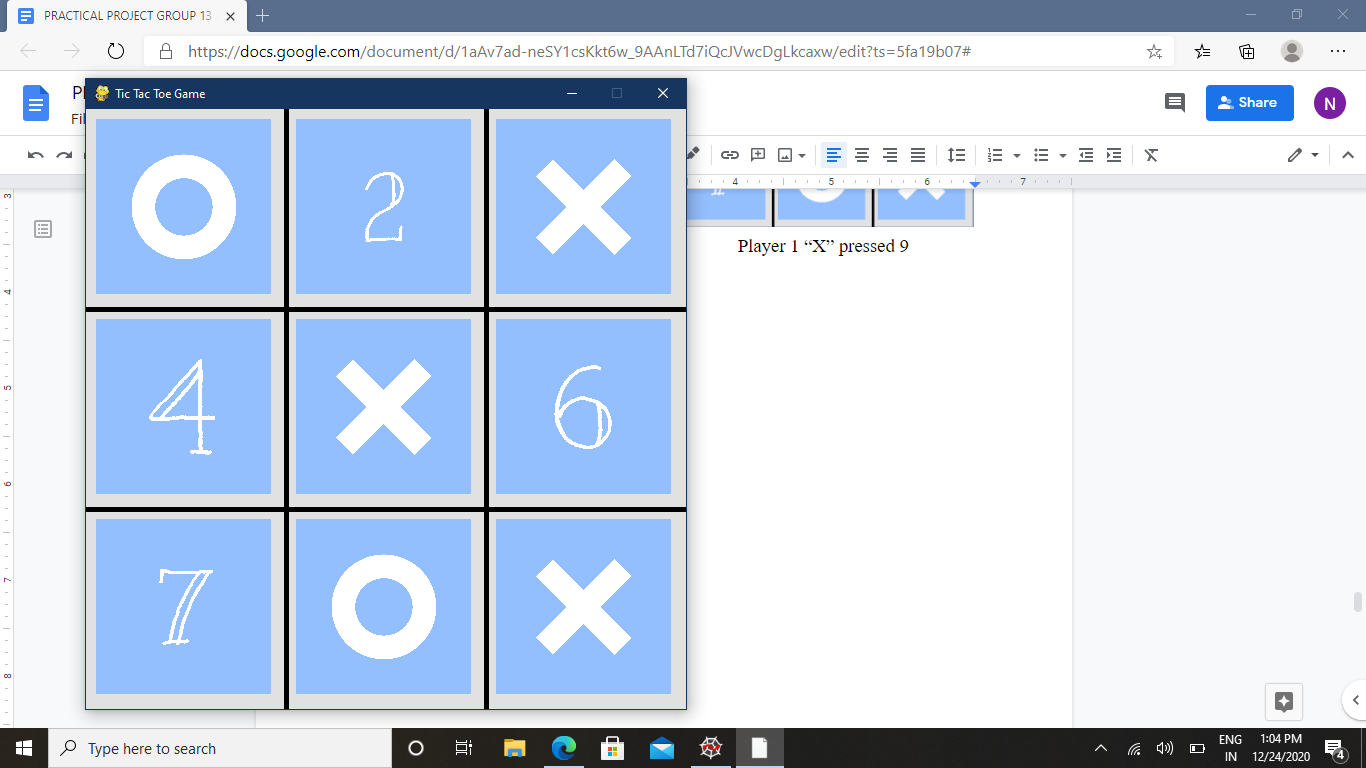
Game :

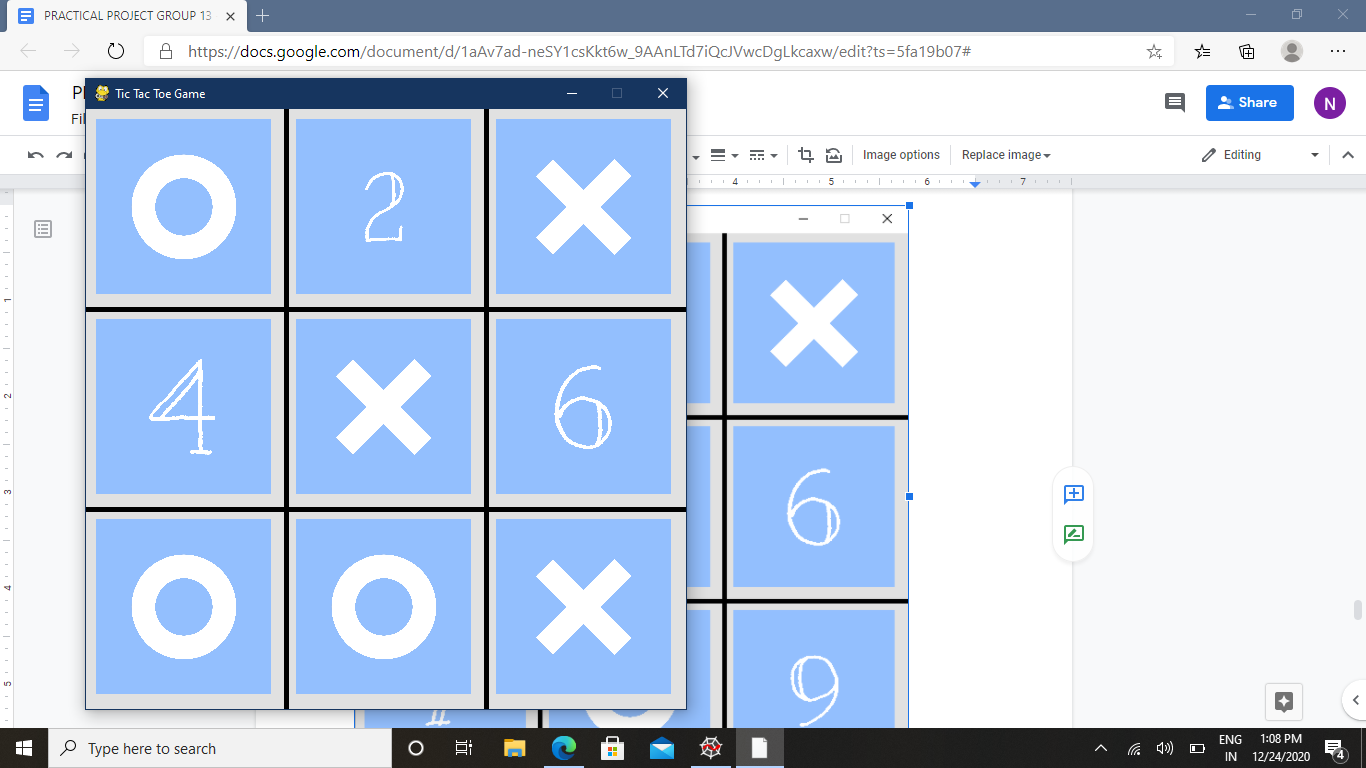
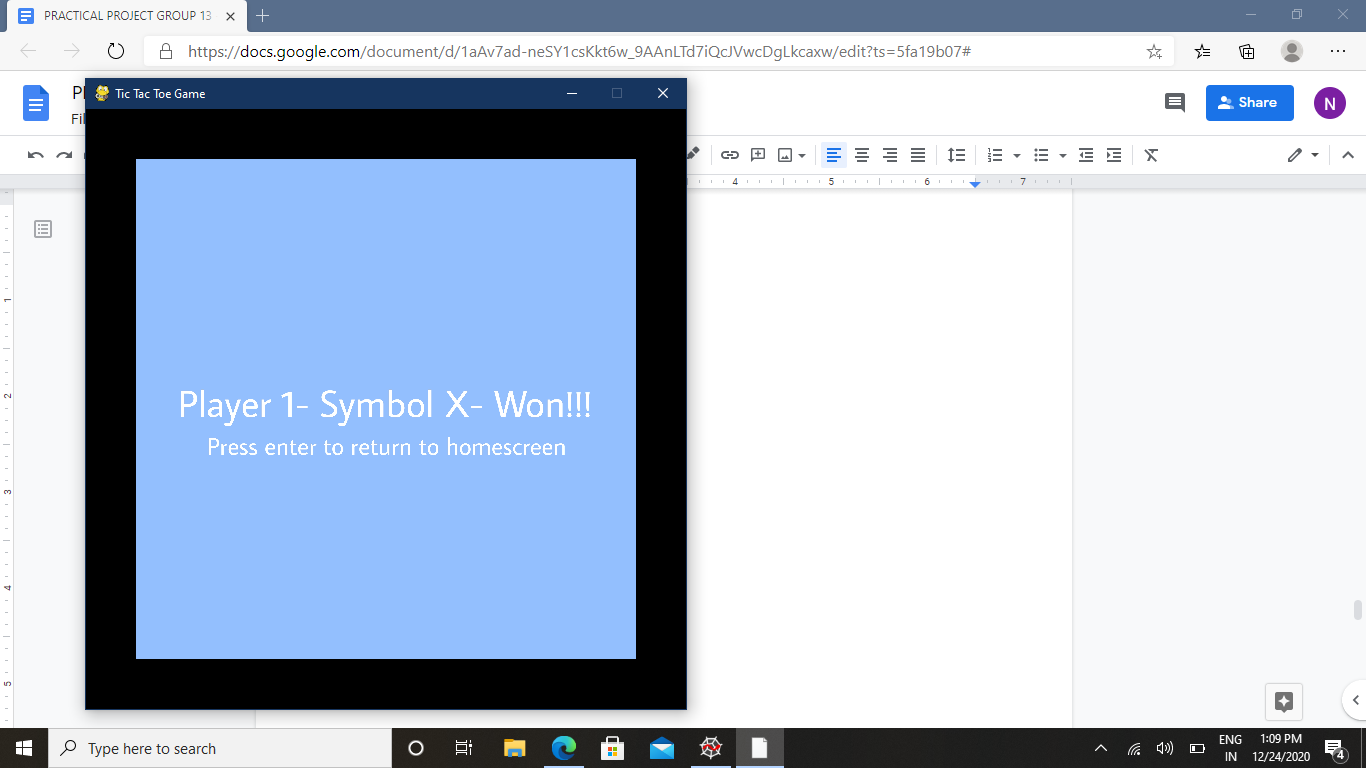
When opened Player 1 “X” pressed 5

Player 2 “O” pressed 8 Player 1 “X” pressed 9

Player 2 “O” pressed 1 Player 1 “X” pressed 3

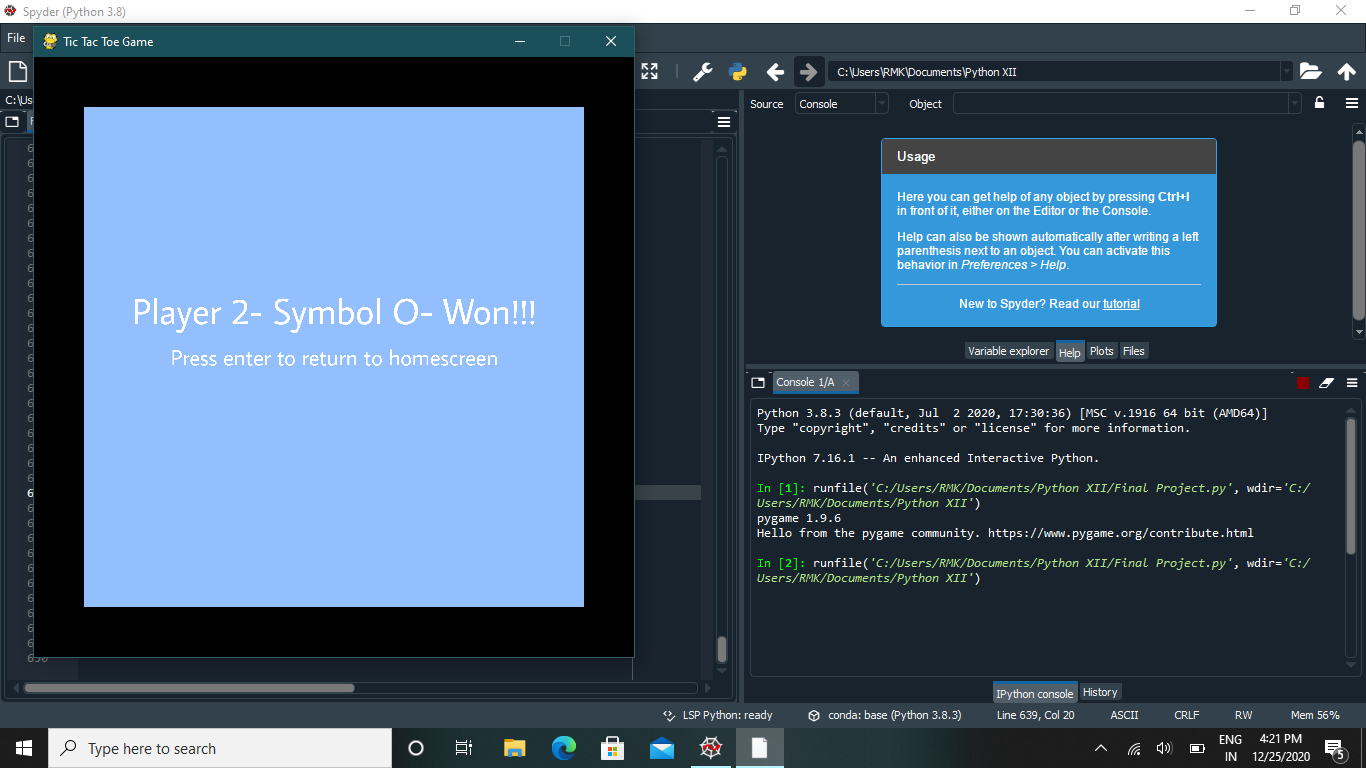
Player 2 “O” pressed 7 When player 1 “X” pressed 6, they won.

Points of user after the game : ( can be seen here as 243 )



Here, the points of the user increased by 10 as they won the game.

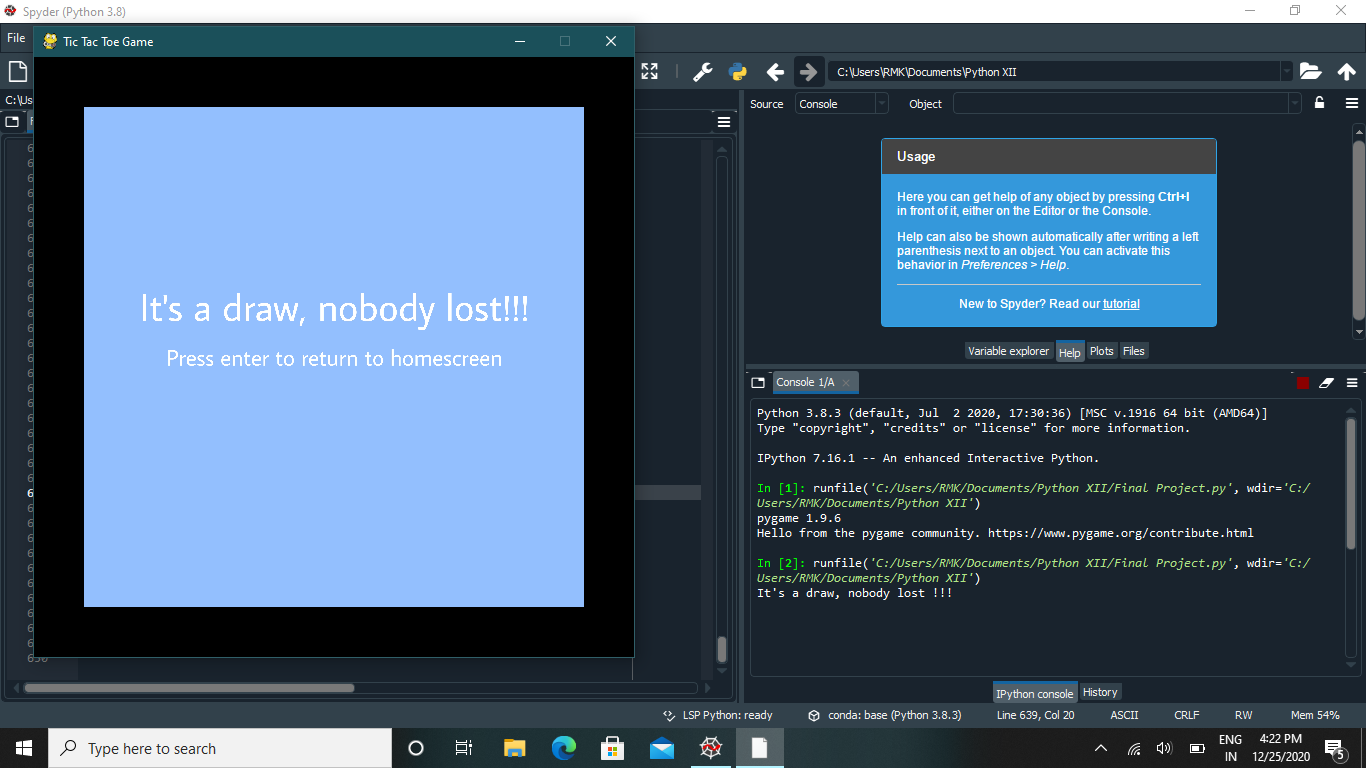
Similarly, if the user ( Player 1 ) loses the game, this shows :



Here, the points of the user don’t change as they lost, therefore before and after the game, the points remain the same :



And, if it’s a draw, this shows :

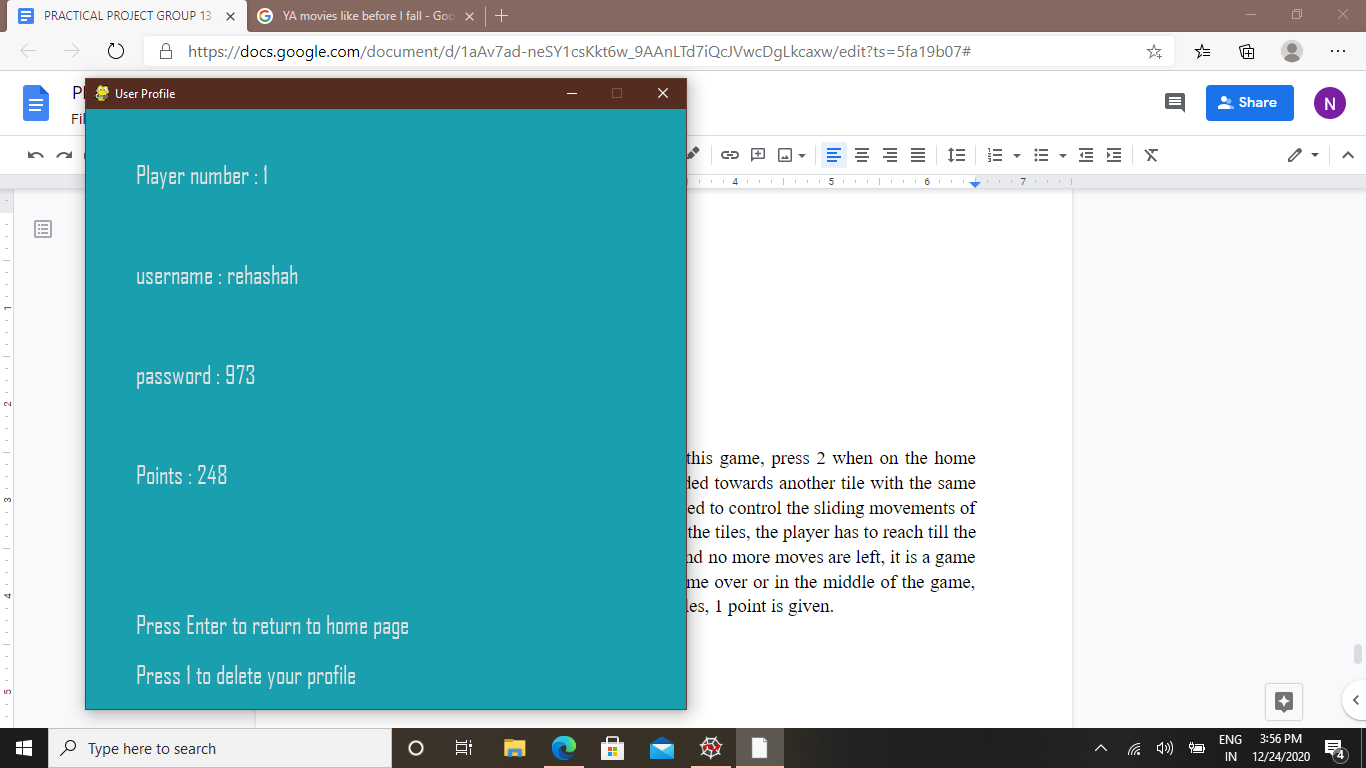


Change in the points of the user in case of draw : ( +5 points )

Before game : ( can be seen here as 243 )



After game : ( can be seen here as 248 )

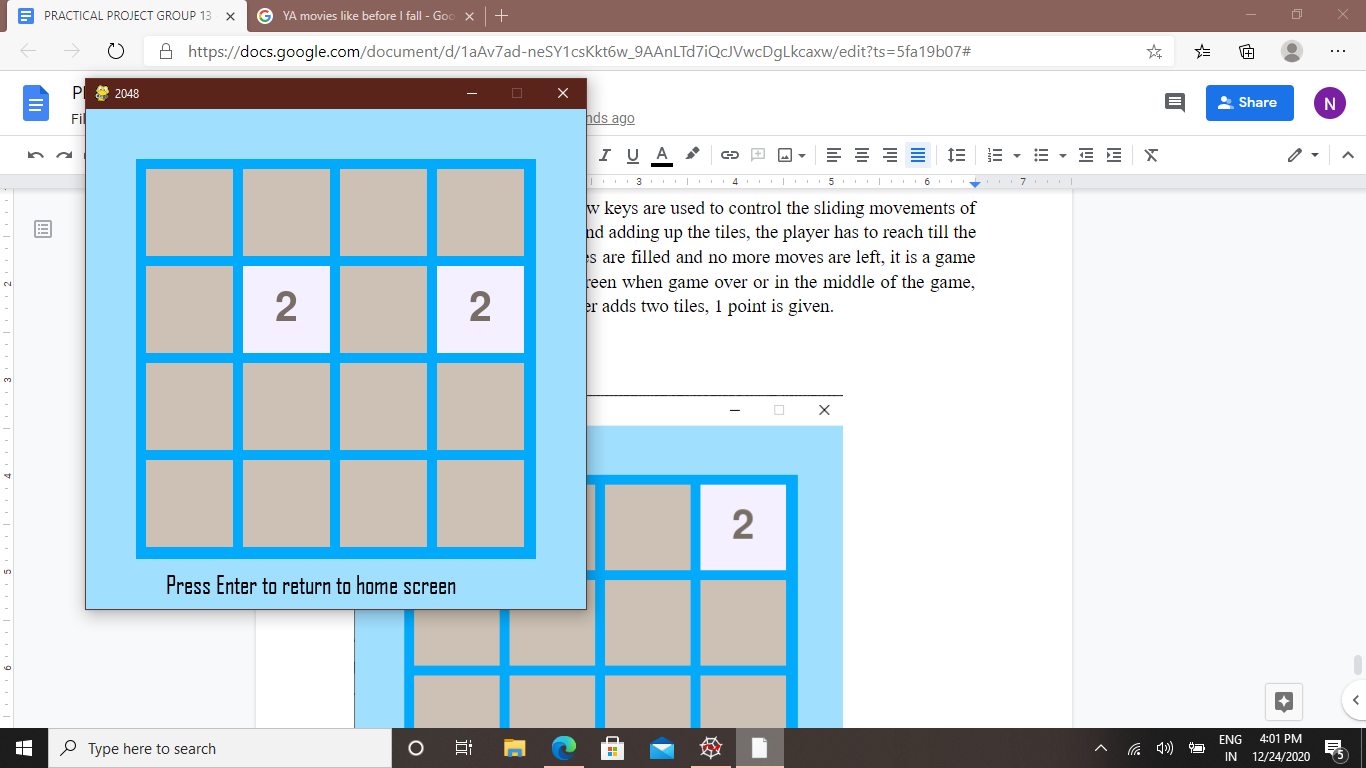
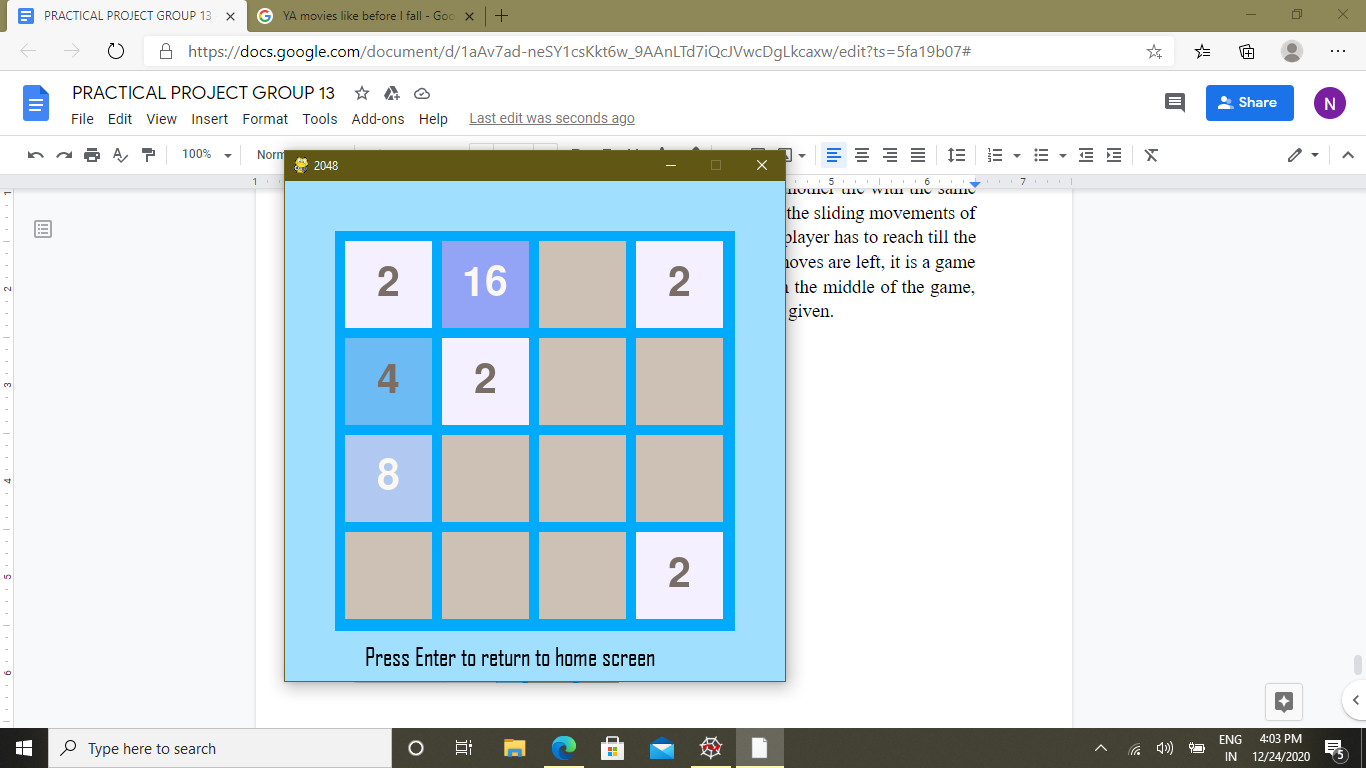


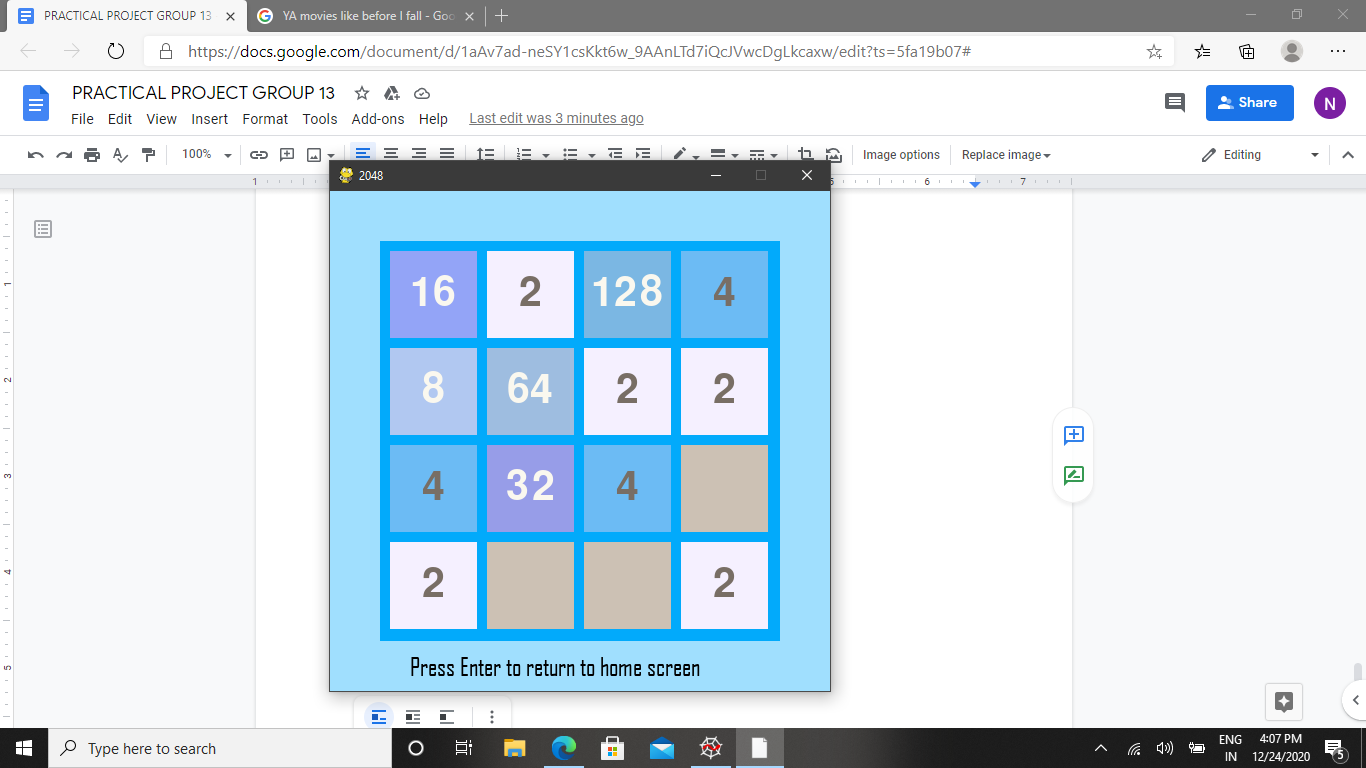
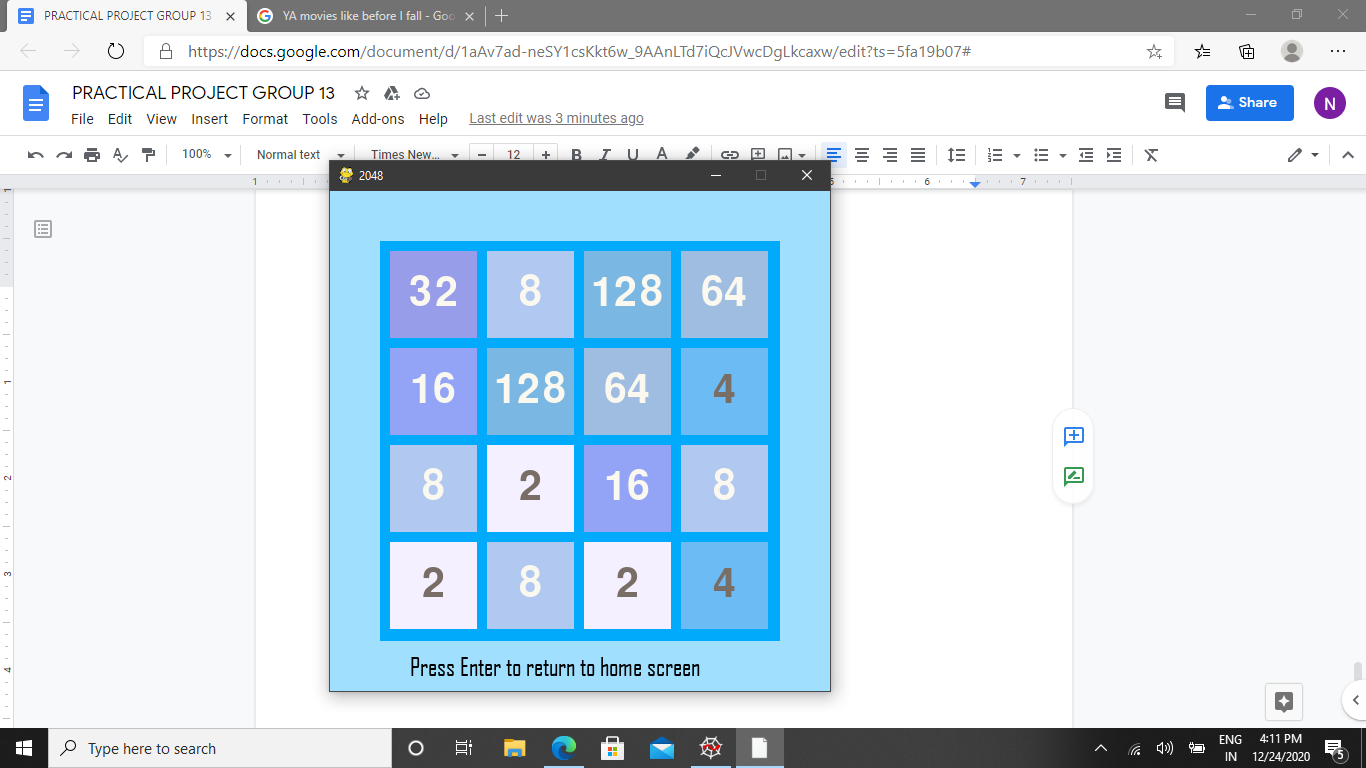
The points of the user went from 243 to 248 as it was a draw, thus increasing by 5.

The 2048 Game:

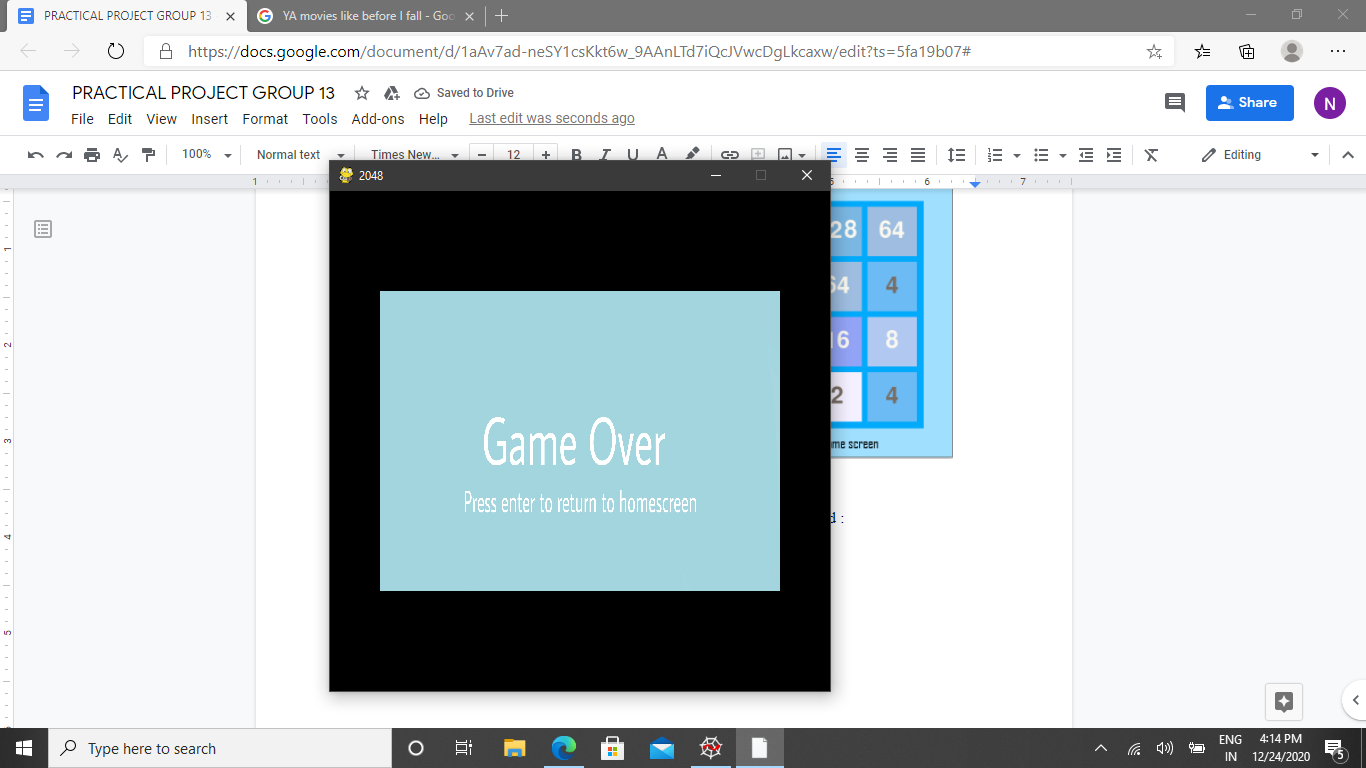
It is a sliding block puzzle game. To play this game, press 2 when on the home page. When a tile with some number is slided towards another tile with the same number, it gets added. The arrow keys are used to control the sliding movements of the tiles. This way, by sliding and adding up the tiles, the player has to reach till the number 2048. When all the tiles are filled and no more moves are left, it is a game over. To return to the home screen when game over or in the middle of the game, press enter. Everytime the player adds two tiles, 1 point is given.

Example of user playing this game :

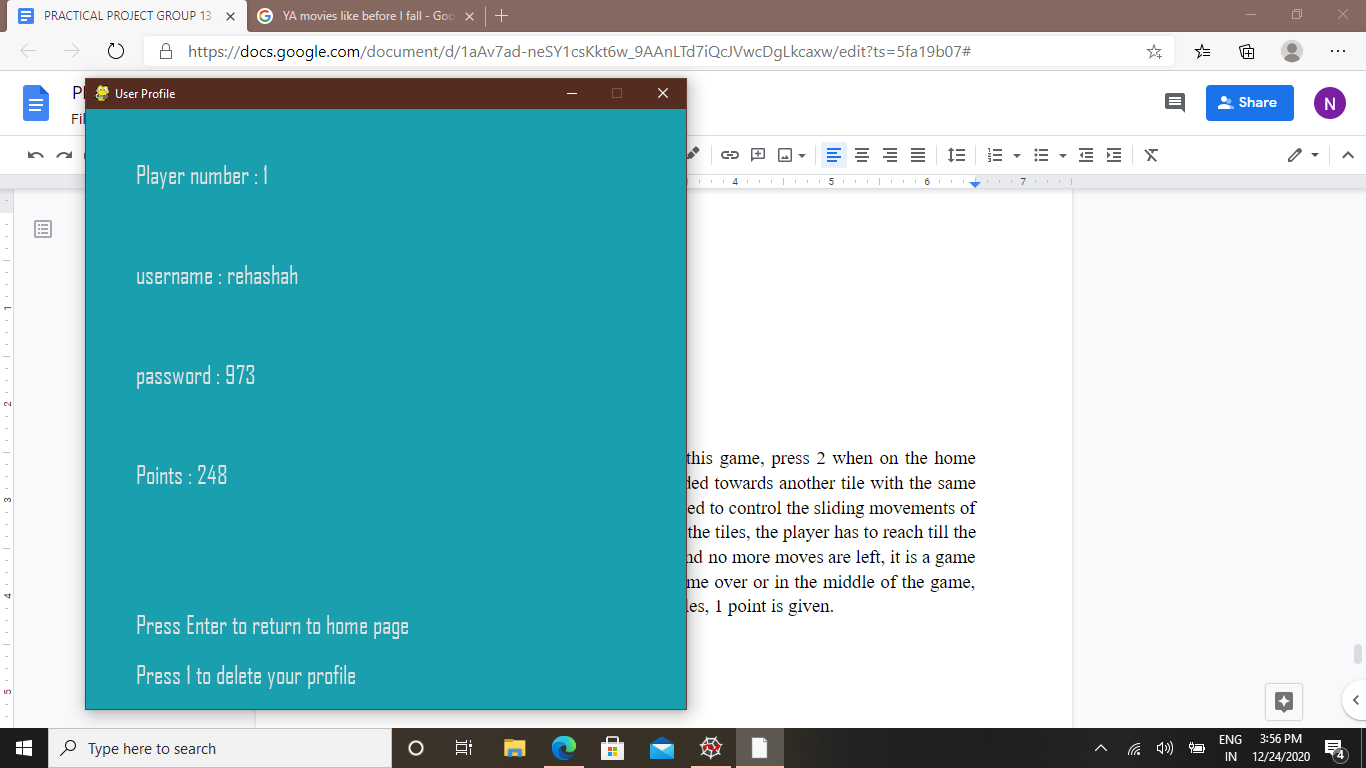
 

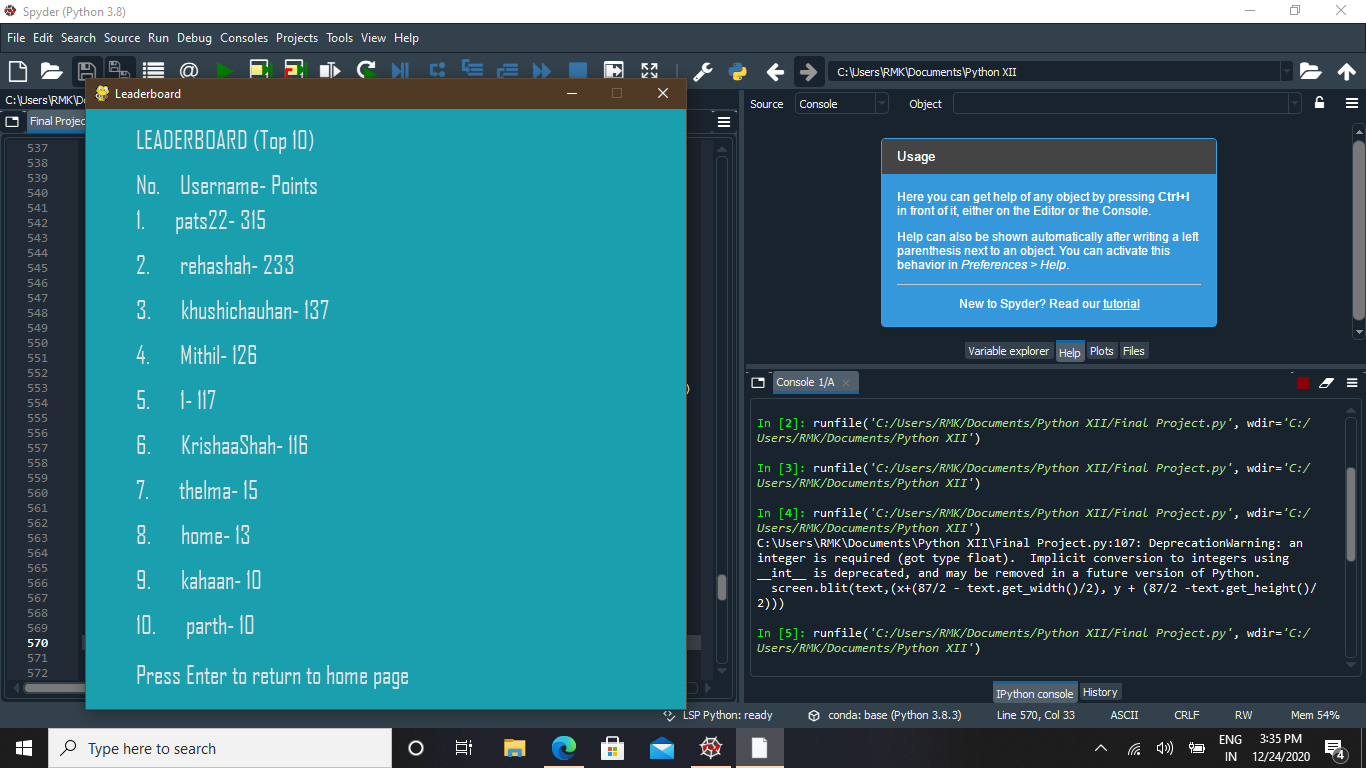
The user can’t do anything more, so they press enter and this gets displayed :



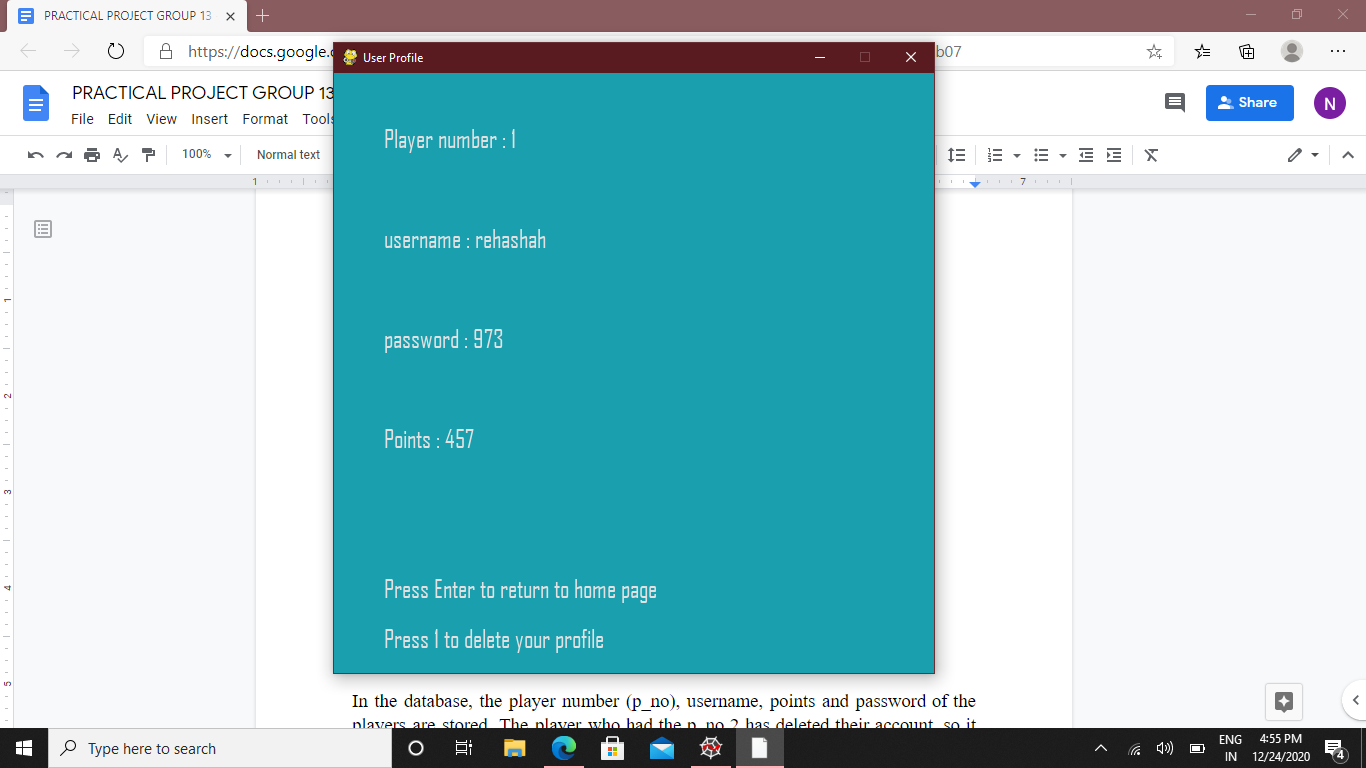
Here, in this example the points of the user increased in this manner :

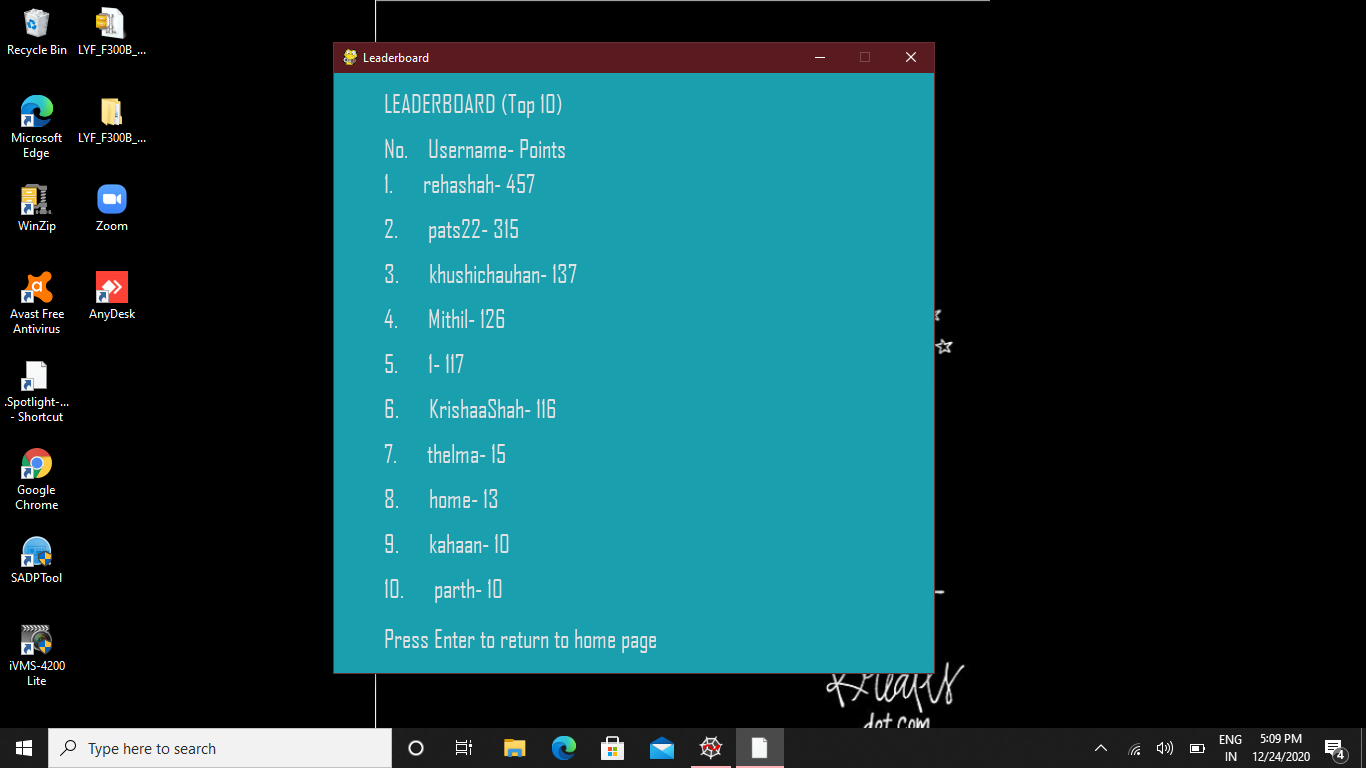
Before playing : ( can see here to be 248 )





After playing : ( can see here to be 457 )





Thus, the player’s points increased by 209. And their rank on the leaderboard also increased from 2nd 1st.

Database stored in MYSQL:

In the database, the player number (p\_no), username, points and password of the players are stored. The players who had the p\_no 3,4,5,6,7 have deleted their account, so it can be seen in this sql table that the numbers 3,4,5,6,7 is missing in between 1 and 9.

