*Sudoku2.py - code*

*# generate random numbers for sudoku*from random import sample  
from tkinter import \*  
import copy  
  
base = 3  
side = base \* base  
*# pattern for a baseline valid solution*def pattern(r, c): return (base \* (r % base) + r // base + c) % side  
*# randomize rows, columns and numbers (of valid base pattern)*def shuffle1(s): return sample(s, len(s))  
  
rBase = range(base)  
rows = [g \* base + r for g in shuffle1(rBase) for r in shuffle1(rBase)]  
cols = [g \* base + c for g in shuffle1(rBase) for c in shuffle1(rBase)]  
nums = shuffle1(range(1, base \* base + 1))  
  
*# produce board using randomized baseline pattern*soln\_board = [[nums[pattern(r, c)] for c in cols] for r in rows]  
board = copy.deepcopy(soln\_board)  
  
*# for line in board: print(line)*squares = side \* side  
empties = squares \* 3 // 4  
for p in sample(range(squares), empties):  
 board[p // side][p % side] = 0  
**'''  
for k in range(9):  
 for m in range(9):  
 print(soln\_board[k][m],end =" ")  
  
 print() '''**def show\_solution():  
 l4 = Label(my\_window,text=**""**,font=(**"Arial Bold"**,12))  
 for i in range(9):  
 for j in range(9):  
 l4[**"text"**]= l4[**"text"**]+ str(soln\_board[i][j]) + **" "** l4[**"text"**] =l4[**"text"**] + **"**\n**"** l4.pack()  
  
  
def check\_solution():  
 result = True  
 temp=0  
 for i,j in index:  
 try:  
 temp = int(l[i][j].get())  
 except ValueError:  
 l3[**"text"**]=**"Enter valid integer values"** if temp != soln\_board[i][j]:  
 result = False  
 break  
 if not result:  
 l3[**"text"**]=**"The solution is invalid Check again"** else:  
 l3[**"text"**]=**"correct solution**\n**"** b2 =Button(my\_window,text=**"Show Solution"**,command=show\_solution)  
 b2.pack()  
  
  
index=[]  
l = [[0 for x in range(9)] for y in range(9)]  
my\_window = Tk()  
my\_window.title(**"Sudoku"**)  
frame1 = Frame(my\_window)  
frame1.pack()  
for i in range(9):  
  
 for j in range(9):  
  
 if board[i][j] != 0:  
 l[i][j] = Label(frame1, text=board[i][j], font=(**"Arial Bold"**, 12))  
 l[i][j].grid(row=i, column=j)  
 else:  
 l[i][j] = Entry(frame1,width=2)  
 l[i][j].grid(row=i, column=j)  
 index.append([i,j])  
  
b1 = Button(my\_window, text=**"Check Solution"**,command=check\_solution)  
b1.pack()  
l3 =Label(my\_window,text=**""**,font=(**"Arial Bold"**,12))  
l3.pack()  
*# my\_window.geometry(300,300)*my\_window.mainloop()

Alarm Clock Code

import time  
from tkinter import \*  
from playsound import playsound  
  
my\_window = Tk()  
my\_window.title(**"Alarm Clock"**)  
  
def display\_time():  
 current\_time = time.strftime(**"%H:%M:%S:%p"**)  
 l1[**'text'**] = current\_time  
 *#l1.configure(text=current\_time)* my\_window.after(1000,display\_time)  
  
def check\_alarm(h,m,d):  
 if d==**"pm"** or d==**"PM"**:  
 h =h +12  
 while True:  
 if(h ==time.localtime().tm\_hour and m == time.localtime().tm\_min):  
 l1[**'text'**] = **"Alarm went on"** playsound(**"/Users/jasmathi/Documents/KloudOne/Python/Class5Assignment/alarm.mp3"**)  
 break  
  
def set\_alarm(event):  
 global str\_day,str\_hr,str\_min  
 res1=**""** str\_time=t1.get()  
 res = **"Alarm has been set to "** + str\_time  
 try:  
 str\_day=str\_time[-2:]  
 str\_min=int(str\_time[3:5])  
 str\_hr=int(str\_time[:2])  
 except:  
 res1 = **"The time should be in hh:mm am or hh:mm pm format"** l2[**'text'**] =res1  
 if res1==**""**:  
 l1[**'text'**]=res  
 check\_alarm(str\_hr, str\_min, str\_day)  
  
  
str\_hr=str\_min=0  
str\_day=**""**l1= Label(my\_window,text =**"Enter the Time"**,font=(**"Arial"**,12),fg=**"blue"**)  
l1.pack(ipady=10)  
t1 =Entry(my\_window,width=10)  
t1.pack(ipady=10)  
l2 =Label(my\_window,text=**""**)  
l2.pack(ipady=10)  
b1 = Button(my\_window, text=**"Set Alarm"**,bg=**"skyblue"**,fg=**"white"**)  
b1.bind(**"<Button-1>"**,set\_alarm)  
b1.pack(ipady=10)  
my\_window.mainloop()

Tic Tac Toe Code

*#Tic Tac Toe Game - Two Player Game*game\_board = [**'\_'**,**'\_'**,**'\_'**,**'\_'**,**'\_'**,**'\_'**,**'\_'**,**'\_'**,**'\_'**]  
game\_over =False  
turn = **'X'**available\_sq = [1,2,3,4,5,6,7,8,9]  
def print\_board():  
 for i in range(0,7,3):  
 print(game\_board[i]+**'**\t**'**+game\_board[i+1]+**'**\t**'**+game\_board[i+2])  
  
  
def push\_board(turn1,sq1):  
 global turn  
 if game\_board[sq1-1] == **'\_'**:  
 game\_board[sq-1] = turn1  
 turn = **'O'** if turn == **'X'** else **'X'** else:  
 print(**"Square Not available.Choose Another Square"**)  
  
def is\_game\_over():  
 if game\_board[0] ==game\_board[1] == game\_board[2]==**'X'** or game\_board[3] == game\_board[4] ==game\_board[5]==**'X'** or game\_board[6]== game\_board[7] ==game\_board[8]==**'X'**or game\_board[0]==game\_board[3]==game\_board[6]==**'X'**or game\_board[1]==game\_board[4]==game\_board[7]==**'X'**or game\_board[2]==game\_board[5]==game\_board[8]==**'X'**or game\_board[0]==game\_board[4]==game\_board[8]==**'X'**or game\_board[2]==game\_board[4]==game\_board[6]==**'X'** :  
 return (True,**'X'**)  
 elif game\_board[0] == game\_board[1] == game\_board[2] == **'O'** or game\_board[3] == game\_board[4] == game\_board[5] == **'O'** or game\_board[6] == game\_board[7] == game\_board[8] == **'O'** or game\_board[0] == game\_board[3] == game\_board[6] == **'O'** or game\_board[1] == game\_board[4] == game\_board[7] == **'O'** or game\_board[2] == game\_board[5] == game\_board[8] == **'O'** or game\_board[0] == game\_board[4] == game\_board[8] == **'O'** or game\_board[2] == game\_board[4] == game\_board[6] == **'O'**:  
 return(True,**'O'**)  
 else:  
 return (False,**''**)  
  
  
def game\_draw():  
 if **'\_'** not in game\_board :  
 return True  
 else:  
 return False  
  
  
print\_board()  
while(game\_over == False):  
 try:  
 sq=int(input(**"Enter a square: "**))  
 except ValueError:  
 print(**"Enter a number between 1-9"**)  
  
 if(sq not in available\_sq):  
 print(**"Invalid Square Number Entered. Try Again."**)  
 continue  
 else:  
 push\_board(turn,sq)  
 print\_board()  
 result =is\_game\_over()  
 if(result[0]):  
 print(**"Game Over"**)  
 print(result[1]+ **" wins the game."**)  
 break  
 if game\_draw():  
 print(**"No more squares available. Game is draw"**)  
 break

Snake Code

import pygame  
import time  
import random  
  
pygame.init()  
  
white = (255, 255, 255)  
yellow = (255, 255, 102)  
black = (0, 0, 0)  
red = (213, 50, 80)  
green = (0, 255, 0)  
blue = (50, 153, 213)  
  
screen\_width = 400  
screen\_height = 400  
  
screen = pygame.display.set\_mode((screen\_width, screen\_height))  
pygame.display.set\_caption(**'Snake Game'**)  
  
clock = pygame.time.Clock()  
  
snake\_block = 10  
snake\_speed = 10  
  
font\_style = pygame.font.SysFont(**"bahnschrift"**, 25)  
score\_font = pygame.font.SysFont(**"comicsansms"**, 35)  
  
def draw\_grid(surface,w,h,r,cs):  
 x=0  
 y=0  
 for \_ in range(r):  
 x=x+cs  
 y=y+cs  
 pygame.draw.line(surface,black,(x,0),(x,h))  
 pygame.draw.line(surface,black,(0,y),(w,y))  
   
   
def Your\_score(score):  
 value = score\_font.render(**"Your Score: "** + str(score), True, green)  
 screen.blit(value, [0, 0])  
  
  
def our\_snake(snake\_block, snake\_list):  
 for x in snake\_list:  
 pygame.draw.rect(screen, blue, [x[0], x[1], snake\_block, snake\_block])  
  
  
def message(msg, color):  
 mesg = font\_style.render(msg, True, color)  
 screen.blit(mesg, [screen\_width // 6, screen\_height // 3])  
  
  
def gameLoop():  
 game\_over = False  
 game\_close = False  
  
 x1 = screen\_width / 2  
 y1 = screen\_height / 2  
  
 x1\_change = 0  
 y1\_change = 0  
  
 snake\_List = []  
 Length\_of\_snake = 1  
  
 foodx = round(random.randrange(0, screen\_width - snake\_block) / 10.0) \* 10.0  
 foody = round(random.randrange(0, screen\_height - snake\_block) / 10.0) \* 10.0  
  
 while not game\_over:  
  
 while game\_close == True:  
 screen.fill(white)  
 message(**"You Lost! C-Play Again Q-Quit"**, red)  
 Your\_score(Length\_of\_snake - 1)  
 pygame.display.update()  
  
 for event in pygame.event.get():  
 if event.type == pygame.KEYDOWN:  
 if event.key == pygame.K\_q:  
 game\_over = True  
 game\_close = False  
 if event.key == pygame.K\_c:  
 gameLoop()  
  
 for event in pygame.event.get():  
 if event.type == pygame.QUIT:  
 game\_over = True  
 if event.type == pygame.KEYDOWN:  
 if event.key == pygame.K\_LEFT:  
 x1\_change = -snake\_block  
 y1\_change = 0  
 elif event.key == pygame.K\_RIGHT:  
 x1\_change = snake\_block  
 y1\_change = 0  
 elif event.key == pygame.K\_UP:  
 y1\_change = -snake\_block  
 x1\_change = 0  
 elif event.key == pygame.K\_DOWN:  
 y1\_change = snake\_block  
 x1\_change = 0  
  
 if x1 >= screen\_width or x1 < 0 or y1 >= screen\_height or y1 < 0:  
 game\_close = True  
 x1 += x1\_change  
 y1 += y1\_change  
 screen.fill(white)  
 pygame.draw.rect(screen, red, [foodx, foody, snake\_block, snake\_block])  
 snake\_Head = []  
 snake\_Head.append(x1)  
 snake\_Head.append(y1)  
 snake\_List.append(snake\_Head)  
 if len(snake\_List) > Length\_of\_snake:  
 del snake\_List[0]  
  
 for x in snake\_List[:-1]:  
 if x == snake\_Head:  
 game\_close = True  
  
 our\_snake(snake\_block, snake\_List)  
 Your\_score(Length\_of\_snake - 1)  
 draw\_grid(screen,screen\_width,screen\_height,40,10)  
 pygame.display.update()  
  
 if x1 == foodx and y1 == foody:  
 foodx = round(random.randrange(0, screen\_width - snake\_block) / 10.0) \* 10.0  
 foody = round(random.randrange(0, screen\_height - snake\_block) / 10.0) \* 10.0  
 Length\_of\_snake += 1  
  
 clock.tick(snake\_speed)  
  
 pygame.quit()  
 quit()  
  
  
gameLoop()