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()
```