*import* pygame  
*import* random  
*import* tkinter *as* tk  
  
*# pyGame initialisation*pygame.init()  
  
*# main window for PyGame*win = pygame.display.set\_mode((750, 600))  
pygame.display.set\_caption("Snakes and Ladders")  
  
*# image asset initialisation*bg = pygame.image.load('assets/board.png')  
char = pygame.image.load('assets/idle.png')  
  
clock = pygame.time.Clock()  
  
  
*def* start\_game():  
 *# tkinter window for getting the names from the user  
 def* set\_name():  
 *# takes the input from teh user and sets the name for that player respectively* player[0].name, player[1].name = e1.get(), e2.get()  
  
 *def* close\_win():  
 *# closes the window* master.destroy()  
  
 master = tk.Tk()  
 master.title("start game")  
 master.geometry("250x100+900+100")  
 tk.Label(master, text="player 1 name").grid(row=0)  
 tk.Label(master, text="player 2 name").grid(row=1)  
  
 e1 = tk.Entry(master)  
 e2 = tk.Entry(master)  
  
 e1.grid(row=0, column=1)  
 e2.grid(row=1, column=1)  
 tk.Button(master, text='Save', command=set\_name).grid(row=3, column=1, sticky=tk.W, pady=4)  
 tk.Button(master, text='start game', command=close\_win).grid(row=3, column=0, sticky=tk.W, pady=4)  
 tk.mainloop()  
  
  
*class* Player(*object*):  
 *# game is designed to be multi-player and object-oriented approach is used to speed things up  
 def \_\_init\_\_*(*self*, x, y):  
 *self*.speed = 51 *# length of one tile on teh screen  
 self*.x, *self*.y = x, y *# x and y co-ordinates for the position  
 self*.direction = 0 *# the direction of the players heading  
 self*.pos = 1 *# position of the tile player is present in  
 self*.name = "player" *# name of the player  
 self*.end\_pos = 1  
  
 *def* movement(*self*, win):  
 *# updating the players position* win.blit(char, (*self*.x, *self*.y))  
  
 *def* change\_dir(*self*):  
 *# changes direction of the player's heading  
 # 0 >> right, 1 >> left  
 if self*.direction == 0:  
 *self*.direction = 1  
 *else*:  
 *self*.direction = 0  
  
  
*def* redraw\_game\_window():  
 *# updates the values on teh screen after every turn* pygame.display.update()  
 *# background update* win.blit(bg, (0, 0))  
 *# player position update  
 for* pla *in range*(*len*(player)):  
 player[pla].movement(win)  
 *# button update* dice\_button.draw(win)  
 start\_game\_button.draw(win)  
 instructions\_button.draw(win)  
  
  
*def* display\_dice\_iterations(count):  
 *# black rectangle to act as a canvas  
 # since the data will be updating after every turn it overlaps the previous data present  
 # and replaces it with updated values* pygame.draw.rect(win, (0, 0, 0), (520, 320, 250, 250), 0)  
 *# setting up 2 fonts for the heading and for the count respectively* font = pygame.font.SysFont('arial', 30)  
 font1 = pygame.font.SysFont('arial', 18)  
 text = font.render("- - Dice count - - ", *True*, (120, 255, 69))  
 text1 = font1.render(" 1 : " + *str*(count[1]), *True*, (120, 255, 69))  
 text2 = font1.render(" 2 : " + *str*(count[2]), *True*, (120, 255, 69))  
 text3 = font1.render(" 3 : " + *str*(count[3]), *True*, (120, 255, 69))  
 text4 = font1.render(" 4 : " + *str*(count[4]), *True*, (120, 255, 69))  
 text5 = font1.render(" 5 : " + *str*(count[5]), *True*, (120, 255, 69))  
 text6 = font1.render(" 6 : " + *str*(count[6]), *True*, (120, 255, 69))  
 *# placing text on the frame* win.blit(text, (520, 320))  
 win.blit(text1, (530, 350))  
 win.blit(text2, (530, 370))  
 win.blit(text3, (530, 390))  
 win.blit(text4, (630, 350))  
 win.blit(text5, (630, 370))  
 win.blit(text6, (630, 390))  
  
  
*def* display\_player\_turn(pla\_num):  
 *# black rectangle to act as a canvas  
 # since the data will be updating after every turn it overlaps the previous data present  
 # and replaces it with updated values* pygame.draw.rect(win, (0, 0, 0), (510, 440, 250, 250), 0)  
 *# setting the font* font = pygame.font.SysFont('arial', 40)  
 text1 = font.render(*str*(player[pla\_num].name)+"'s turn", *True*, (120, 255, 69))  
 *# placing text on the frame* win.blit(text1, (510, 440))  
  
  
*def* display\_player\_positions():  
 *# black rectangle is needed to act as a canvas  
 # since the data will be updated after every turn it overlaps the previous data present  
 # and replaces it with updated values* pygame.draw.rect(win, (0, 0, 0), (0, 500, 500, 300), 0)  
 pygame.draw.rect(win, (0, 0, 0), (400, 500, 550, 250), 0)  
 *# setting the font* font = pygame.font.SysFont('arial', 35)  
 text1 = font.render(*str*(player[0].name) + "'s position : " + *str*(player[0].pos), *True*, (120, 255, 69))  
 text2 = font.render(*str*(player[1].name) + "'s position : " + *str*(player[1].pos), *True*, (120, 255, 69))  
 *# placing text on the frame* win.blit(text1, (10, 520))  
 win.blit(text2, (400, 520))  
  
  
*def* move\_player(n):  
 *# takes player number as argument and moves the player accordingly  
 # print(player[player\_number].name , player[player\_number].pos)  
 def* change\_player():  
 *# changes player  
 global* player\_number  
 *if* player\_number == 1:  
 player\_number = 0  
 *else*:  
 player\_number = 1  
  
 *def* dice\_roll():  
 *# returns a random number between 1 to 6 same as that of a dice* val = random.randint(1, 6)  
 *# print(val)  
 return* val  
  
 *def* win\_check(x):  
 *# x co-ordinate values with corresponding dice roll to win* positions = {306: 6, 255: 5, 204: 4, 153: 3, 102: 2, 51: 1}  
 plapos = positions[x]  
 *# as the player has run out of options for moving, the movements need to be restricted  
 if* roll > plapos:  
 *# print("try again")* player[player\_number].pos -= 1  
 *# position is calculated by summing up the dice rolls,  
 # so, the invalid move dice rolls should be subtracted  
 else*:  
 player[player\_number].x -= player[player\_number].speed  
  
 *if* player[player\_number].x == 0:  
 *print*("player won")  
 player[player\_number].pos += (roll-1)  
  
 *def* snake\_check():  
 *pass  
  
 # def ladder\_check():  
 # # print(player[player\_number].pos, player[player\_number].name)  
 # print(roll)  
 # if player[player\_number].pos == 3:  
 # player[player\_number].x -= player[player\_number].speed\*2  
 # player[player\_number].y -= player[player\_number].speed\*4  
 # player[player\_number].pos = 41  
 #  
 # if player[player\_number].pos == 6:  
 # player[player\_number].x += player[player\_number].speed\*1  
 # player[player\_number].y -= player[player\_number].speed\*2  
 # player[player\_number].pos = 27  
 #  
 # if player[player\_number].pos == 11:  
 # player[player\_number].y -= player[player\_number].speed\*3  
 # player[player\_number].pos = 50  
 #  
 # if player[player\_number].pos == 36:  
 # player[player\_number].x -= player[player\_number].speed\*1  
 # player[player\_number].y -= player[player\_number].speed\*2  
 # player[player\_number].pos = 57  
 # # if player[player\_number].pos == 3:  
 # # player[player\_number].x += player[player\_number].speed  
 # # if player[player\_number].pos == 3:  
 # # player[player\_number].x += player[player\_number].speed  
  
 def* movement(n):  
 *# moves the player by one step  
 if* player[n].y == 1 *and* player[n].x <= 306:  
 win\_check(player[n].x)  
  
 *else*:  
 *if* player[n].direction == 0:  
 *if* player[n].x == 459:  
 player[n].y -= player[n].speed-2  
 player[n].change\_dir()  
 *# print(player[n].x)  
 # print(player[n].y)  
 else*:  
 player[n].x += player[n].speed  
 *# print(player[n].x)  
 # print(player[n].y)  
  
 elif* player[n].direction == 1:  
 *if* player[n].x == 0:  
 player[n].y -= player[n].speed  
 player[n].change\_dir()  
 *# print(player[n].x)  
 # print(player[n].y)  
 else*:  
 player[n].x -= player[n].speed  
 *# print(player[n].x)  
 # print(player[n].y)* roll = dice\_roll()  
 *# making the movements in one block at at time, so the position can be noted and evaluated  
 for* ik *in range*(roll):  
 movement(n)  
  
 *# saving the position of the player by summing up the dice rolls* player[n].pos += roll  
 *# saving each dice roll in dict* dice\_count[roll] += 1  
  
 *#ladder\_check()  
 # print("player name :", player[n].name, "player position >> ", player[n].pos)  
 # changing player turn* change\_player()  
  
  
*class* Button(*object*):  
 *# a class made for easy creations of buttons  
 # takes color(hex values) x and y co-ordinates, width, height and the text  
 def \_\_init\_\_*(*self*, color, x, y, width, height, text=''):  
 *self*.color = color  
 *self*.x = x  
 *self*.y = y  
 *self*.width = width  
 *self*.height = height  
 *self*.text = text  
  
 *# for placing it on the screen  
 def* draw(*self*, win):  
 pygame.draw.rect(win, *self*.color, (*self*.x, *self*.y, *self*.width, *self*.height), 0)  
 font = pygame.font.SysFont('arial', 40)  
 text = font.render(*self*.text, 1, (0, 0, 0))  
 win.blit(text, (*self*.x + (*self*.width / 2 - text.get\_width() / 2), *self*.y + (*self*.height / 2 - text.get\_height() / 2)))  
  
 *# tells us if the mouse pointer is over the element or not  
 def* is\_over(*self*, pos):  
 *if self*.x < pos[0] < *self*.x + *self*.width:  
 *if self*.y < pos[1] < *self*.y + *self*.height:  
 *return True  
 return False  
  
 # can be used to update the color of the button  
 def* update\_color(*self*, color):  
 *self*.color = color  
  
  
*# initializer statements*player = [1, 2]  
dice\_count ={1: 0, 2: 0, 3: 0, 4: 0, 5: 0, 6: 0}  
*# setting the players up in the board  
for* i *in range*(*len*(player)):  
 player[i] = Player(0, 450)  
  
*# the three buttons that are present on the screen*dice\_button = Button((96, 69, 96), 505, 5, 240, 100, "roll dice")  
start\_game\_button = Button((96, 69, 96), 505, 110, 240, 100, "start game")  
instructions\_button = Button((96, 69, 96), 505, 215, 240, 100, "instructions")  
  
run = *True*direction, player\_number = 1, 0  
  
*# main loop  
while* run:  
 clock.tick(54)  
 *for* event *in* pygame.event.get():  
 pos = pygame.mouse.get\_pos()  
 *if* event.type == pygame.QUIT:  
 run = *False # exits us out of the loop thus ending the game  
  
 if* event.type == pygame.MOUSEBUTTONDOWN:  
 *if* dice\_button.is\_over(pos):  
 move\_player(player\_number)  
 *if* start\_game\_button.is\_over(pos):  
 start\_game()  
 *# checks if the mouse button is down and then the position of the mouse pointer  
 # if the pointer location falls inside of the button the functions gets called  
  
 # hover functionality for the buttons  
 if* dice\_button.is\_over(pygame.mouse.get\_pos()):  
 dice\_button.update\_color((200, 2, 225))  
 *else*:  
 dice\_button.update\_color((120, 72, 5))  
  
 *if* start\_game\_button.is\_over(pygame.mouse.get\_pos()):  
 start\_game\_button.update\_color((200, 2, 225))  
 *else*:  
 start\_game\_button.update\_color((120, 72, 5))  
  
 *if* instructions\_button.is\_over(pygame.mouse.get\_pos()):  
 instructions\_button.update\_color((200, 2, 225))  
 *else*:  
 instructions\_button.update\_color((120, 72, 5))  
  
 *# redraws game window* redraw\_game\_window()  
  
 display\_dice\_iterations(dice\_count)  
 display\_player\_turn(player\_number)  
 display\_player\_positions()  
  
pygame.quit()