```
#basic library imports
import tkinter as tk
import tkinter.messagebox
from time import sleep
from random import choice
from PIL import ImageTk, Image


from settings import *
from board import *



class Coin:


    def __init__(self, master, x, y, color, path_list, flag):
        self.canvas = master
        self.curr_x = x
        self.curr_y = y
        self.home_x = x
        self.home_y = y
        self.color = color
        self.curr_index = -1
        #A picture that works with Tkinter. Anywhere that Tkinter expects an image object can use this.
        #When an image is an RGBA picture, pixels with an alpha value of 0 are considered translucent.
```

```
        self.coin = ImageTk.PhotoImage(Image.open('./assets/{}.png'.format(color)))
        self.img =  self.canvas.create_image(x, y, anchor=tk.NW, image=self.coin)
        self.canvas.tag_bind(self.img, '<1>', self.moveCoin)
        self.disable = True
        self.path_list = path_list
        self.flag = flag
        self.win = 0
        self.pad_x = 0


    #this function is responsible for handling the movement of tokens of each
player
    def moveCoin(self, event):


        if self.disable:
            return
        #this conditional will handle the rollinf of dice
        roll = Dice.roll
        if len(roll) == 0:
            return


        if roll[-1] == 6:
            tkinter.messagebox.showerror('Error', 'You got 6, Please Roll Again')
            return


        if len(roll) != 0 :
```

```
            n = len(self.path_list)
            max_moves = n - self.curr_index - 1
            if max_moves < roll[0]:
                return
        #this is to handle the movement of tokens in Ludo Game in Python
        check = (False, 0, 0)
        congrats = False
        if self.is_at_home():
            #this conditional is for the number 6 on dice.
            if 6 in roll:
                check = self.can_attack(0)
                self.canvas.coords(self.img, self.path_list[0][0] + 4 + self.pad_x,
self.path_list[0][1] + 4)
                self.curr_x = self.path_list[0][0]
                self.curr_y = self.path_list[0][1]
                self.curr_index = 0
                Dice.remove_by_index(6)
        #this elif block will check if the token is able to attack on any other token or
not
        else:
            check = self.can_attack(self.curr_index + roll[0])
            for i in range(roll[0] - 1):
                self.curr_index += 1
                self.canvas.coords(self.img, self.path_list[self.curr_index][0] + 4,
self.path_list[self.curr_index][1] + 4)
                self.curr_x = self.path_list[self.curr_index][0]
```

```
                self.curr_y = self.path_list[self.curr_index][1]
                self.canvas.update()
                sleep(0.05)


            self.curr_index += 1
            self.canvas.coords(self.img, self.path_list[self.curr_index][0] + 4 +
self.pad_x, self.path_list[self.curr_index][1] + 4)
            self.curr_x = self.path_list[self.curr_index][0]
            self.curr_y = self.path_list[self.curr_index][1]
            if check[0]:
                colors[check[1]][check[2]].goto_home()


            self.canvas.update()
            sleep(0.05)
            Dice.remove()


            if self.curr_index == len(self.path_list) - 1:
                self.win = 1
                tkinter.messagebox.showinfo('INFO','!! Congratulations !!\nPlease Roll
Dice Again')
                congrats = self.congratulations()
             #now that if a user is able to kill a token , then another chance will be
given to the players
            if check[0]:
                tkinter.messagebox.showinfo('INFO','You killed another coin! Now you
get another chance.\nPlease Roll Dice Again')
```

```
            congrats = self.congratulations()

    if self.is_player_won():
        tkinter.messagebox.showinfo('INFO','{} Wins'.format(self.color.title()))
        position.append(self.player.title())
        Dice.roll = []
        Dice.set(self.flag)

    if self.is_gameover():
        root.quit()

    if not check[0] and not congrats:
        if len(Dice.roll):
            Dice.check_move_possibility()
        self.next_turn()



def congratulations(self):
    Dice.update_state()
    Dice.set(self.flag - 1)

    return True

def change_state(self, flag):
    if flag == self.flag:
```

```
            self.disable = False
        else:
            self.disable = True


    #this is to check if the token is at home or not
    def is_at_home(self):
        return self.curr_x == self.home_x and self.curr_y == self.home_y


    def check_home(self):
        count = 0
        for goti in colors[self.flag]:
            if goti.is_at_home():
                count += 1


        return count
    #this is to check the status of player, whether he/she is a winnner or Not
    #this is done by using the conditionals
    #here we will update the goti.win variable by 1 whenever the user's token
reaches home
    #hen it is 4 the player will be declared as the winner of Ludo Game in Python
    def is_player_won(self):
        reached = 0
        for goti in colors[self.flag]:
            if goti.win:
                reached += 1
```

```
        return reached is 4


    #this is to check whether the Ludo Game in Python is over or not
    def is_gameover(self):
        color_reached = 0


        for i in range(4):
            game = 0
            for color in colors[i]:
                if color.win:
                    game += 1
            if game is 4:
                color_reached += 1


        if color_reached is 3:
            tkinter.messagebox.showinfo('Game Over', '\n\n1. {}\n\n2. {}\n\n3.
{}'.format(*position))
        else:
            return False
        return True
    #this below code will check whether the player will be able to attack other
token or not
    def can_attack(self, idx):
        max_pad = 0
        count_a = 0
```

```
x = self.path_list[idx][0]
y = self.path_list[idx][1]
for i in range(4):
    for j in range(4):
        if colors[i][j].curr_x == x and colors[i][j].curr_y == y:
            if colors[i][j].pad_x > max_pad:
                max_pad = colors[i][j].pad_x
            count_a += 1

if not self.path_list[idx][2]:
    for i in range(4):
        count = 0
        jdx = 0
        for j in range(4):
            if (colors[i][j].curr_x == x and colors[i][j].curr_y == y
                and colors[i][j].color != self.color):
                count += 1
                jdx = j

        if count is not 0 and count is not 2:
            self.pad_x = max_pad + 4
            return (True, i, jdx)

if count_a is not 0:
    self.pad_x = max_pad + 4
```

```
        else:
            self.pad_x = 0
        return (False, 0, 0)
    #will handle the event when user will go to home
    def goto_home(self):
        self.canvas.coords(self.img, self.home_x, self.home_y)
        self.curr_x = self.home_x
        self.curr_y = self.home_y
        self.curr_index = -1

    #will be used to handle the next turn event
    def next_turn(self):
        if len(Dice.roll) == 0:
            Dice.set(self.flag)
    #will be used to set the player's name
    def set_playername(self, player):
        self.player = player

#this is to handle the outcomes of the dice
class Dice:

    chance = 0
    roll = []
    append_state = False
```

```
    @classmethod
    def rolling(cls):
        temp = choice(range(1, 9))
        if temp > 6:
            temp = 6


        if len(cls.roll) == 0 or cls.roll[-1] == 6 or cls.append_state:
            cls.roll.append(temp)
            cls.append_state = False
        #here we made a tuple of 6 images of dice and any one will be selected in a random manner
        dice = {
            1: 'de1.png',
            2: 'de2.png',
            3: 'de3.png',
            4: 'de4.png',
            5: 'de5.png',
            6: 'de6.png',
        }.get(cls.roll[-1], None)


        img = ImageTk.PhotoImage(Image.open('./assets/{}'.format(dice)))
        image_label = tk.Label(ludo.get_frame(), width=100, height=100, image=img, bg=Color.CYAN)
        image_label.image = img
        image_label.place(x=250, y=300)
```

```
        roll_label = tk.Label(ludo.get_frame(), text='{}'.format(' | '.join([str(x) for x in
cls.roll])),
                              font=(None, 20), width=30, height=3, borderwidth=3,
relief=tk.RAISED)
        roll_label.place(x=100, y=200)


    @classmethod
    def start(cls):
        Dice.rolling()
        if cls.roll.count(6) >= 3:
            if [cls.roll[-1], cls.roll[-2], cls.roll[-3]] == [6, 6, 6]:
                for i in range(3):
                    Dice.remove_by_index(6)


            if cls.roll == []:
                Dice.update_panel()
                return
        Dice.check_move_possibility()


    @classmethod
    def update_panel(cls):
        root.update()
        sleep(0.5)
        Dice.set(cls.chance)
        cls.roll = []
```

```
    @classmethod
    def set(cls, flag):
        flag += 1
        cls.chance = flag
        if flag == 4:
            cls.chance = flag = 0
        if colors[cls.chance][0].is_player_won():
            Dice.set(cls.chance)
        else:
            for i in range(4):
                for j in range(4):
                    colors[i][j].change_state(flag)


            next_label = tk.Label(ludo.get_frame(), text='{} turn'.format(turn[flag]),
font=(None, 20), width=30, height=3,
                                  borderwidth=3, relief=tk.SUNKEN)
            next_label.place(x=100, y=100)


            roll_label = tk.Label(ludo.get_frame(), text='ROLL PLEASE', font=(None,
20), width=30, height=3, borderwidth=3, relief=tk.RAISED)
            roll_label.place(x=100, y=200)


            img = ImageTk.PhotoImage(Image.open('./assets/trans.png'))
            image_label = tk.Label(ludo.get_frame(), width=100, height=100,
image=img, bg=Color.CYAN)
            image_label.image = img
```

```
        image_label.place(x=250, y=300)


@classmethod
def remove(cls):
    Dice.roll.pop(0)


@classmethod
def remove_by_index(cls, ex):
    del cls.roll[cls.roll.index(ex)]


@classmethod
def update_state(cls):
    cls.append_state = True


@classmethod
def check_move_possibility(cls):
    check_1 = 0
    check_2 = 0
    for goti in colors[cls.chance]:
        if goti.is_at_home():
            check_1 += 1
        else:
            max_moves = len(goti.path_list) - goti.curr_index - 1
            if max_moves < cls.roll[0]:
                check_2 += 1
```

```
        if 6 not in cls.roll:
            if check_1 is 4 or check_1 + check_2 is 4:
                Dice.update_panel()
        else:
            if check_2 is 4:
                Dice.update_panel()



def align(x, y, color, path_list, flag):
    container = []
    for i in range(2):
        test = Coin(ludo.get_canvas(), x, y + i*2*Board.SQUARE_SIZE, color=color,
path_list=path_list, flag=flag)
        container.append(test)
    for i in range(2):
        test = Coin(ludo.get_canvas(), x + 2*Board.SQUARE_SIZE, y +
i*2*Board.SQUARE_SIZE, color=color, path_list=path_list, flag=flag)
        container.append(test)


    return container


#the functionality to begin the game is handled here
def startgame():
    for i in range(4):
        if players[i].get():
```

```
            turn[i] = players[i].get()
    for i in range(4):
        for j in range(4):
            colors[i][j].set_playername(turn[i])


    start_label = tk.Label(ludo.get_frame(), text='! START ! Let\'s Begin with {}'.format(turn[0]), font=(None, 20),
                           width=30, height=3, borderwidth=3, relief=tk.SUNKEN)
    start_label.place(x=100, y=100)
    top.destroy()


#this is to create the second window in our Ludo Game in Python.
#to add the functionality and design to our nickname window we used the label() function
def create_enterpage():
    #in order to set the font, width, height etc we used the varios parameters of label()
    enter_label = tk.Label(top, text='Enter Your Nickname!', font=(None, 20), width=30, height=3,
                           borderwidth=3, relief=tk.RAISED)
    enter_label.place(x=20, y=20)


    enter_button = tk.Button(top, text='Enter', command=startgame, width=15, height=2)
    enter_button.place(x=230, y=500)


    for i in range(2):
```

```
        temp = tk.Entry(top, width=15)
        temp.place(x=87, y=220 + i*180)
        players.append(temp)

    for i in range(2):
        temp = tk.Entry(top, width=15)
        temp.place(x=387, y=400 - i*180)
        players.append(temp)

    global greenimg, redimg, blueimg, yellowimg

    #this is the code to render the four different images of our dice
    greenimg = ImageTk.PhotoImage(Image.open('./assets/green2.png'))
    green_label = tk.Label(top, image=greenimg)
    #in order to set the postion of various token on nickname window we used x
and y
    green_label.place(x=107, y=130)

    redimg = ImageTk.PhotoImage(Image.open('./assets/red2.png'))
    red_label = tk.Label(top, image=redimg)
    red_label.place(x=107, y=310)

    blueimg = ImageTk.PhotoImage(Image.open('./assets/blue2.png'))
    blue_label = tk.Label(top, image=blueimg)
    blue_label.place(x=407, y=310)
```

```
    yellowimg = ImageTk.PhotoImage(Image.open('./assets/yellow2.png'))
    yellow_label = tk.Label(top, image=yellowimg)
    yellow_label.place(x=407, y=130)


#now to handle the closing of the game, we have used the destroy() functionality
#this will basically destroy all the widgets in Ludo Game in Python
def on_closing():
    if tkinter.messagebox.askokcancel("Quit", "Do you want to quit the game? If you want to continue the game, press Enter in the Nickname window"):
        top.destroy()
        root.destroy()


def on_closingroot():
    if tkinter.messagebox.askokcancel("Quit", "Do you want to quit the game?"):
        root.destroy()


#this is where the set up of tkinter window is handled
players = []
root = tk.Tk()
width = root.winfo_screenwidth()
height = root.winfo_screenheight()
root.geometry('{}x{}'.format(width, height))
root.title('Ludo')
```

```
ludo = LudoBoard(root)

ludo.create()



turn = ['Green', 'Red', 'Blue', 'Yellow']

position = []

colors = []

colors.append(align(2.1*Board.SQUARE_SIZE, 2.1*Board.SQUARE_SIZE,
color='green', path_list=path.green_path, flag=0))

colors.append(align(2.1*Board.SQUARE_SIZE, 11.1*Board.SQUARE_SIZE,
color='red', path_list=path.red_path, flag=1))

colors.append(align(11.1*Board.SQUARE_SIZE, 11.1*Board.SQUARE_SIZE,
color='blue', path_list=path.blue_path, flag=2))

colors.append(align(11.1*Board.SQUARE_SIZE, 2.1*Board.SQUARE_SIZE,
color='yellow', path_list=path.yellow_path, flag=3))



for i in range(4):

    for j in range(4):

        colors[i][j].change_state(0)



button = tk.Button(ludo.get_frame(), text='ROLL', command=Dice.start, width=20,
height=2)

button.place(x=210, y=470)



#this is the message that will be displayed whenever the user will start the game

welcome_msg = ''' Welcome Champs let's get into the game of LUDO :-) \n

        Rules of the game:
```

- The players roll a six-sided die in turns and can advance any of their coins on the track by the number of steps as displayed by the dice.\n

- Once you get a six in a dice throw, you have to roll the dice again, and must use all scores while making the final selection of what coins to move where.\n

- If you get a six three times in a row, your throws are reset and you will lose that chance.\n

- The coin can advance in the home run only if it reaches exactly inside the home pocket, or moves closer to it through the home run.

For example, if the coin is four squares away from the home pocket and the player rolls a five, he must apply the throw to some other coin. \

However, if you roll a two, you can advance the coin by two squares and then it rests there until the next move.\n

Enjoy the game and have fun.

Best of luck

'''

#we used the Tkinter's messagebox.showinfo() function to render the message stored in the above variable

tkinter.messagebox.showinfo('Welcome', welcome_msg)

#once the rules window is displayed, a new window to set the names of the player will be opened

top = tk.Toplevel(root)

top.geometry('600x600')

top.title('Nickname')

top.protocol("WM_DELETE_WINDOW", on_closing)

root.protocol("WM_DELETE_WINDOW", on_closingroot)

create_enterpage()

```
root.mainloop()
```