import pygame

import sys

from Board import \*

import tkinter as tk

from tkinter import \*

pygame.init()

BG = pygame.image.load("images\BackGround.png")

def get\_font(size): # Returns Press-Start-2P in the desired size

    return pygame.font.Font("images\Font.ttf", size)

class Button():

    def \_\_init\_\_(self,screen ,image, x\_pos, y\_pos, text\_input , font , base\_color , hovering\_color):

        self.image = image

        self.x\_pos = x\_pos

        self.y\_pos = y\_pos

        self.font = font

        self.base\_color = base\_color

        self.hovering\_color = hovering\_color

        self.rect = self.image.get\_rect(center=(self.x\_pos, self.y\_pos))

        self.text\_input = text\_input

        self.text = font.render(self.text\_input, True, self.base\_color)

        if self.image is None:

            self.image = self.text

        self.text\_rect = self.text.get\_rect(center=(self.x\_pos, self.y\_pos))

        self.screen=screen

    def update(self):

        if self.image is not None:

            self.screen.blit(self.image, self.rect)

        self.screen.blit(self.text, self.text\_rect)

    def checkForInput(self):

        if self.rect.collidepoint(pygame.mouse.get\_pos()):

            return True

        return False

    def changeColor(self):

        if self.rect.collidepoint(pygame.mouse.get\_pos()):

            self.text = self.font.render(self.text\_input, True, self.hovering\_color)

        else:

            self.text = self.font.render(self.text\_input, True, self.base\_color)

button\_surface = pygame.image.load("images\Button.png")

button\_surface = pygame.transform.scale(button\_surface, (400, 150))

button\_surfaceOptions = pygame.transform.scale(button\_surface, (600, 150))

class MainMenu():

    def \_\_init\_\_(self):

        pygame.init()

        self.screen = pygame.display.set\_mode((800, 800))

        self.newBG = pygame.transform.scale(BG, (800, 800))

        self.gameover=False

        self.players=4

    def options(self):

        while True:

            self.screen.fill("white")

            OPTIONS\_TEXT = get\_font(40).render("This is the OPTIONS screen.", True, "Green")

            OPTIONS\_RECT = OPTIONS\_TEXT.get\_rect(center=(550, 260))

            self.screen.blit(OPTIONS\_TEXT, OPTIONS\_RECT)

            for event in pygame.event.get():

                if event.type == pygame.QUIT:

                    pygame.quit()

                    sys.exit()

                if event.type == pygame.MOUSEBUTTONDOWN:

                        self.main\_menu()

                        return

            pygame.display.update()

    def main\_menu(self):

        pygame.display.set\_caption("Menu")

        menu\_text = get\_font(70).render("Main Menu",True , "#b68f40")

        menu\_rect = menu\_text.get\_rect(center=(400, 100))

        PLAY\_BUTTON = Button(self.screen, button\_surface, 400, 250, "PLAY", get\_font(75), "#d7fcd4", "White")

        OPTIONS\_BUTTON = Button(self.screen, button\_surfaceOptions, 400, 450, "OPTIONS", get\_font(75), "#d7fcd4", "White")

        QUIT\_BUTTON = Button(self.screen, button\_surface, 400, 650, "QUIT", get\_font(75), "#d7fcd4", "White")

        while True:

                self.screen.blit(self.newBG,(0,0))

                self.screen.blit(menu\_text, menu\_rect)

                menu\_text=get\_font(70).render("Main Menu",True , "#b68f40")

                menu\_rect = menu\_text.get\_rect(center=(400, 100))

                if menu\_rect.collidepoint(pygame.mouse.get\_pos()):

                    menu\_text = get\_font(70).render("Main Menu", True, "White")

                else:

                    menu\_text=get\_font(70).render("Main Menu",True , "#b68f40")

                for button in [PLAY\_BUTTON,OPTIONS\_BUTTON,QUIT\_BUTTON]:

                    button.changeColor()

                    button.update()

                for event in pygame.event.get():

                    if event.type == pygame.QUIT:

                        pygame.quit()

                        sys.exit()

                    if event.type == pygame.MOUSEBUTTONDOWN:

                        if PLAY\_BUTTON.checkForInput():

                            pygame.quit()

                            True

                            play(self,self.players)

                            return

                        if OPTIONS\_BUTTON.checkForInput():

                            t=animate\_win("how many players are there","1","2","3","4")

                            self.players=t.get()

                        if QUIT\_BUTTON.checkForInput():

                            pygame.quit()

                            sys.exit()

                PLAY\_BUTTON.update()

                OPTIONS\_BUTTON.update()

                QUIT\_BUTTON.update()

                pygame.display.update()

class animate\_win:

    def \_\_init\_\_(self,msg,button1="",button2="",button3="",button4=""):

        self.data=0

        reg\_font = ("Verdana", 22)

  #popup = tk.Tk()

        self.popup = Tk()

        frame = Frame(self.popup)

        frame.pack(side="bottom")

        self.popup.minsize(800, 600) #

        self.popup.wm\_title("Result")

        text = tk.Label(self.popup, text=msg, font=reg\_font)

        text.pack(side="top",padx=30,pady=30)

        if button1!="":

            b1 = tk.Button(frame, text=button1, padx=5,pady=5, command = self.set1)

            b1.config(height = 3, width = 2,padx=5,pady=5)

            b1.pack(side=LEFT,padx=30,pady=30)

        if button2!="":

            b1 = tk.Button(frame, text=button2, padx=5,pady=5, command = self.set2)

            b1.config(height = 3, width = 2,padx=5,pady=5)

            b1.pack(side=LEFT,padx=30,pady=30)

        if button3!="":

            b1 = tk.Button(frame, text=button3, padx=5,pady=5, command = self.set3)

            b1.config(height = 3, width = 2,padx=5,pady=5)

            b1.pack(side=LEFT,padx=30,pady=30)

        if button4!="":

            b1 = tk.Button(frame, text=button4, padx=5,pady=5, command = self.set4)

            b1.config(height = 3, width = 2,padx=5,pady=5)

            b1.pack(side=LEFT,padx=30,pady=30)

        self.popup.mainloop()

    def set1(self):

        self.data=1

        self.popup.destroy()

    def set2(self):

        self.data=2

        self.popup.destroy()

    def set3(self):

        self.data=3

        self.popup.destroy()

    def set4(self):

        self.data=4

        self.popup.destroy()

    def get(self):

        return self.data

from Board import \*

from BoardGame import \*

mainMenu=MainMenu()

mainMenu.main\_menu()

from tkinter import Tk, Canvas, Frame, Label ,Button

from PIL import Image, ImageTk

import sys

import pygame

import random

from MiniGame\_One import MiniGame\_One

from MiniGame\_Two import MiniGame\_Two

from input\_text import InputText

from flappy\_bird import flappyBird

boardsize=24

qsize=6

board\_code =["&","x","\_","+","\_","x"]\*4

ROWS,COLS = 7,7

WIDTH, HEIGHT = 100,100

SQUARE\_SIZE = 100

SQUARE\_SIZE\_X = WIDTH // COLS

SQUARE\_SIZE\_Y = WIDTH // COLS

# Define color constants

RED = (255,0,0)

WHITE = (255,255,255)

BLACK = (0,0,0)

BLUE = (0,0,255)

class Player:

    image\_path\_temp =""

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

        if tempplayernumber==1:

            self.pos =0

            image\_path\_temp = "images\playerOne.png"

            self.grid=(0,0)

        if tempplayernumber==2:

            self.pos =6

            image\_path\_temp = "images\playerTwo.png"

            self.grid=(6,0)

        if tempplayernumber==3:

            self.pos =12

            image\_path\_temp = "images\playerThree.png"

            self.grid=(6,6)

        if tempplayernumber==4:

            self.pos =18

            image\_path\_temp = "images\playerFour.png"

            self.grid=(0,6)

        self.coins=0

        self.min\_roll=1

        self.max\_roll=6

        self.movesToWin=boardsize

        self.playnum=tempplayernumber

        original\_image\_temp = Image.open(image\_path\_temp)

        resized\_image\_temp = original\_image\_temp.resize((SQUARE\_SIZE-50, SQUARE\_SIZE-50))

        self.image =ImageTk.PhotoImage(resized\_image\_temp)

    #function position

    def setgrid(self):

        #roll=1

        global boardsize, qsize # Add player\_positions to the list of global variables

        #self.pos=(self.pos+roll)%boardsize

        fliped=(self.pos>=(qsize\*2))

        temp=self.pos%(qsize\*2)

        if fliped:

            temp=(qsize\*2)-temp

        fi,si=0,0

        if temp < qsize:

            fi=0

            si=temp%qsize

        elif temp == qsize\*2:

            fi=qsize

            si=qsize

        else:

            fi=temp%qsize

            si=qsize

        if fliped:

            self.grid = (fi,si)  #player\_four position update

        else:

            self.grid = (si,fi)

    def move(self,x):

        self.pos= (self.pos+x)%boardsize

        self.movesToWin-=x

        self.setgrid()

    def position(self):

        self.x =self.grid[0]   # x position

        self.y =self.grid[1]    #y position

    def lose\_coins(self,x):

        if self.coins>=x:

            self.coins-=x

        else:

            self.move(self.coins-x)

            self.coins=0

    def roll(self):

        return random.randrange(self.min\_roll,self.max\_roll)

    def draw(self,rect,player):  #load the image on the board on the giving position by the function position

        if player==1:

            rect.create\_image(50, 50, anchor="nw", image=self.image)

        if player==2:

            rect.create\_image(0, 50, anchor="nw", image=self.image)

        if player==3:

            rect.create\_image(0, 0, anchor="nw", image=self.image)

        if player==4:

            rect.create\_image(50, 0, anchor="nw", image=self.image)

class EndTurnButton(Button):

    def \_\_init\_\_(self,root,my\_button\_move):

        super().\_\_init\_\_(root,text="End Turn ",command=self.on\_button\_click,width=10,height=3)

        self.my\_button\_move=my\_button\_move

    def on\_button\_click(self):

        self.grid\_remove()

        self.my\_button\_move.grid(row=ROWS // 2, column=COLS // 2)

"""

class PlayButton(Button):

    def \_\_init\_\_(self, root):

        # Call the constructor of the base class (Button)

        super().\_\_init\_\_(root, text="Play ", command=self.on\_button\_click,width=10,height=3)

        self.gameWon = False

        self.button\_move = MyButtonMove(root,False,text="Move")

        self.button\_move.grid(row=ROWS//2, column=COLS//2)

        self.button\_move = MyButtonMove(root,False,text="Move")

        self.button\_move.grid(row=(ROWS//2)+1, column=COLS//2)

        self.endTurn = False

        self.row=0

        self.col=0

    def on\_button\_click(self):

        pass

    def getendTurn(self):

        return self.endTurn

class MyButtonMove(Button):

    def \_\_init\_\_(self, root,endTurn,text="Play "):

        # Call the constructor of the base class (Button)

        super().\_\_init\_\_(root, text=text, command=self.on\_button\_click,width=10,height=3)

        #self.player\_positions = player\_positions

        self.endTurn = endTurn

        #self.button\_move = MyButtonMove(root,player\_positions,False,text="Move")

        self.button\_End = EndTurnButton(root,self)

    def getEndButton(self):

        return self.button\_End

    def on\_button\_click(self):

        roll=random.randrange(1,6)

        #roll=1

        # (roll)

        Ptemp=board.get\_current\_player()

        Ptemp.move(roll)

        if board\_code[Ptemp.pos]=="x":

            Ptemp = red\_land(Ptemp)

        Ptemp.setgrid()

        board.move\_player(Ptemp.grid[1],Ptemp.grid[0],board.get\_current\_player\_index()-1)

        board.set\_current\_player(Ptemp)

        if board\_code[Ptemp.pos]=="&":

            Ptemp = blue\_land()

        board.update\_players()"""

class ButtonRoll(Button):

    def \_\_init\_\_(self, root,www,text="Play "):

        super().\_\_init\_\_(root, text=text, command=self.on\_button\_click,width=10,height=3)

    def on\_button\_click(self):

        board.rollFun()

class ButtonMove(Button):

    def \_\_init\_\_(self, root,www,text="Play "):

        super().\_\_init\_\_(root, text=text, command=self.on\_button\_click,width=10,height=3)

    def on\_button\_click(self):

        board.moveFun()

class ButtonEnd(Button):

    def \_\_init\_\_(self, root,www,text="Play "):

        super().\_\_init\_\_(root, text=text, command=self.on\_button\_click,width=10,height=3)

    def on\_button\_click(self):

        board.endFun()

class Board:

    def \_\_init\_\_(self,menu,num\_of\_players\_in=3):

        self.num\_of\_players=num\_of\_players\_in

        self.menu=menu

        self.PLAYERS=[]

        for x in range(self.num\_of\_players):

            self.PLAYERS.append(Player(x+1))

        self.board = [] # Initialize an empty list to represent the game board.

        #self.player\_positions = [(0, 0), (6,0), (6, 6), (0,6)]

        self.squares = [[None for \_ in range(COLS)] for \_ in range(ROWS)]

        self.create\_board(0,0)

        self.draw\_squares()

        #root.bind("<Key>", self.on\_key\_press)

        self.gameWon = False

        #self.buttonPlay= MyButtonMove(root,False,"Move",self)

        #self.buttonPlay.grid(row=ROWS//2, column=COLS//2)

        self.buttonTop= ButtonRoll(root,1,text="roll")

        self.buttonTop.grid(row=ROWS//2, column=COLS//2)

        self.stage=1

        self.current\_player\_index = 1

        self.roll=0

        self.label = Label(root, text="Turn Player 1", bg="#D3D3D3",padx=10, pady=10)

        self.labelTwo = Label(root, text="Blue", bg="#D3D3D3",padx=10, pady=10)

        self.labelWin = Label(root, text="You Have Won ", bg="#D3D3D3",padx=10, pady=10)

        self.sideLabel= Label(root)

        self.labelPlayer(self.current\_player\_index)

    def stage1(self):

        self.buttonTop= ButtonRoll(root,1,text="roll")

        self.buttonTop.grid(row=ROWS//2, column=COLS//2)

        self.buttonBot.destroy()

        self.sideLabel.destroy()

        self.stage=1

    def stage2(self):

        self.buttonTop.destroy()

        self.buttonTop= ButtonMove(root,1,text="move")

        self.buttonTop.grid(row=ROWS//2, column=COLS//2)

        self.buttonBot= ButtonEnd(root,1,text="end")

        self.buttonBot.grid(row=(ROWS//2)+1, column=COLS//2)

        self.sideLabel=Label(root, text=str(self.roll), bg="#D3D3D3",padx=10, pady=10)

        self.sideLabel.grid(row=3, column=2,columnspan=2,sticky='nw')

        self.stage=2

    def stage3(self):

        self.buttonTop= ButtonEnd(root,1,text="end")

        self.buttonTop.grid(row=ROWS//2, column=COLS//2)

        self.buttonBot.destroy()

        self.stage=3

        self.sideLabel.destroy()

    def stageChange(self):

        if self.stage==1:

            self.stage2()

        elif self.stage==2:

            self.stage3()

        elif self.stage==3:

            self.stage1()

    def rollFun(self):

        self.roll=self.PLAYERS[self.current\_player\_index-1].roll()

        #self.roll=60

        self.stage2()

    def moveFun(self):

        self.PLAYERS[self.current\_player\_index-1].move(1)

        board.move\_player()

        self.roll-=1

        self.sideLabel.config(text=str(self.roll))

        #self.sideLabel=Label(root, text=str(self.roll), bg="#D3D3D3",padx=10, pady=10)

        if self.roll==0:

            self.stage3()

    def endFun(self):

        Ptemp=self.PLAYERS[self.current\_player\_index-1]

        if board\_code[Ptemp.pos]=="x":

            Ptemp = red\_land(Ptemp)

        elif board\_code[Ptemp.pos]=="&":

            Ptemp = blue\_land(Ptemp)

        elif board\_code[Ptemp.pos]=="+":

            Ptemp = green\_land(Ptemp)

        board.move\_player()

        board.update\_players()

        self.stage1()

    def get\_current\_player\_index(self):

        return self.current\_player\_index

    def get\_current\_player(self):

        return self.PLAYERS[self.current\_player\_index-1]

    def set\_current\_player(self,temp):

        self.PLAYERS[self.current\_player\_index-1]=temp

        if  self.PLAYERS[self.current\_player\_index-1].movesToWin<=0:

            self.buttonPlay.destroy()

            #pygame.quit()

            #sys.exit()

    def swap\_current\_player(self):

        # Toggle the current player's index (0 to 1, 1 to 0)

        self.current\_player\_index = 2 if self.current\_player\_index == 1 else 1

    def update\_players(self):

        if self.PLAYERS[self.current\_player\_index-1].movesToWin<=0:

            self.buttonBot.destroy()

            self.buttonTop.destroy()

            self.sideLabel.destroy()

            #self.labelWin.grid(row=3, column=3,columnspan=3,sticky='nw')

            self.redraw\_board()

            self.\_\_init\_\_(self.menu)

            self.open\_mainMenuAgain()

        self.current\_player\_index += 1

        if self.current\_player\_index>self.num\_of\_players:

            self.current\_player\_index = 1

        self.labelPlayer(self.current\_player\_index)

    def getPlayButton(self):

        return self.buttonPlay

    def draw\_squares(self):

        for rows in range(ROWS):

            for col in range (COLS):

                if  col == 0 or (rows < 1) or col == COLS-1 or rows>ROWS-2:

                    if col in range (rows%2==0,COLS,2):

                        colorBg="red"

                    else:

                        colorBg="black"

                else:

                    colorBg="white"

                rect = Canvas(root, width=SQUARE\_SIZE, height=SQUARE\_SIZE, bg=colorBg)

                rect.grid(row=rows, column=col)

                self.squares[rows][col]=rect

                frame = Frame(root, bg="yellow", width=100, height=100)

                frame.grid(row=ROWS//2, column=COLS//2)

        self.redraw\_board()

    def create\_board(self,row,col):

        for row in range(ROWS):

            self.board.append([])

            for col in range(COLS):

                if (row, col) in [(0, 0), (0, 6), (6, 6), (6, 0)]:

                    # (str(row)+" "+str(col))

                    for x in self.PLAYERS:

                        self.board[row].append(x)

                else:

                    self.board[row].append(0)

    def move\_player(self):

        self.player\_positions=[]

        for x in self.PLAYERS:

            self.player\_positions.append(x.grid)

        self.redraw\_board()

    def redraw\_board(self,fillerBg="white"):

        # Redraw the board with updated player position

        for row in range(ROWS):

            for col in range(COLS):

                if  col == 0 or (row < 1) or col == COLS-1 or row>ROWS-2:

                    if col in range (row%2==0,COLS,2):

                        colorBg="red"

                    else:

                        colorBg="black"

                else:

                    colorBg="white"

                if (col==0 and row==0)or(col==0 and row==6)or(col==6 and row==0)or(col==6 and row==6):

                    colorBg="blue"

                if ((col+row==3)and(col==0 or row==0)) or ((col+row==9)and(col==6 or row==6)):

                    colorBg="green"

                rect = self.squares[row][col]

                rect.config(bg=colorBg)

                rect.delete("all")

                #player 1

                count=0

                for x in self.PLAYERS:

                    canvas\_widget=self.squares[x.grid[1]][x.grid[0]]

                    if count ==0:

                        canvas\_widget.create\_image(0, 0, anchor="nw", image=x.image)

                    if count ==1:

                        canvas\_widget.create\_image(50, 0, anchor="nw", image=x.image)

                    if count ==2:

                        canvas\_widget.create\_image(50, 50, anchor="nw", image=x.image)

                    if count ==3:

                        canvas\_widget.create\_image(0, 50, anchor="nw", image=x.image)

                    count=count+1

    def labelPlayer(self,currentPlayer):

            if currentPlayer==1:

                self.label.grid(row=1, column=1,columnspan=2,sticky='nw')

                self.labelTwo.grid(row=1, column=1,columnspan=2,sticky='sw')

                self.labelTwo.config(text="Blue")

            elif currentPlayer==2:

                self.label.grid(row=1, column=5,columnspan=2,sticky='nw')

                self.labelTwo.grid(row=1, column=5,columnspan=2,sticky='sw')

                self.labelTwo.config(text="Green")

            elif currentPlayer==3:

                self.label.grid(row=5, column=5,columnspan=2,sticky='nw')

                self.labelTwo.grid(row=5, column=5,columnspan=2,sticky='sw')

                self.labelTwo.config(text="Red")

            elif currentPlayer==4:

                self.label.grid(row=5, column=1,columnspan=2,sticky='nw')

                self.labelTwo.grid(row=5, column=1,columnspan=2,sticky='sw')

                self.labelTwo.config(text="Yellow")

            self.label.config(text="Turn Player "+str(self.current\_player\_index))

    def open\_mainMenuAgain(self):

        #board.reset\_game()

        self.menu.\_\_init\_\_()

        root.destroy()

        self.menu.main\_menu()

        #root.quit()

def blue\_land(player):

    rand=random.randrange(1,4)

    def blue\_land\_1():

        rand=random.randrange(1,4)

        itt= InputText("what player would you like to move back")

        aceptable=["1","2","3","4"]

        if itt.getInput() in aceptable:

            Ptemp=board.PLAYERS[int(itt.getInput())]

            #board.move\_player(Ptemp.grid[1],Ptemp.grid[0],int(itt.getInput()))

            board.PLAYERS[int(itt.getInput())-1].move(-5)

            board.move\_player()

            #board.PLAYERS[int(itt.getInput())]=Ptemp

        else:

            i= InputText("invalid input "+itt.getInput())

        return player

    def blue\_land\_2():

        temp\_coins=player.coins

        player.move(temp\_coins-5)

        return player

    def blue\_land\_3():

        rand=random.randrange(1,11)

        itt= InputText("make a guess x if guess is under random number move forward x if its over move backwards x")

        aceptable=["1","2","3","4","5","6","7","8","9","10"]

        if itt.getInput() in aceptable:

            x= int(itt.getInput())

            if x>rand:

                itt= InputText("you guesed over the random number was "+str(rand)+" moving back "+str(x))

                player.move(-x)

            else:

                itt= InputText("you guesed under the random number was "+str(rand)+" moving forward "+str(x))

                player.move(x)

        else:

            i= InputText("invalid input "+itt.getInput())

        return player

    if rand==1:

        t = InputText("move other")

        return blue\_land\_1()

    elif rand==2:

        t = InputText("purse check")

        return blue\_land\_2()

    elif rand==3:

        t = InputText("test your luck minigame")

        return blue\_land\_3()

    else:

        return player

def red\_land(player):

    rand=random.randrange(1,4)

    def red\_land\_1():

        rand=random.randrange(1,4)

        if rand==1:

            game = MiniGame\_One()

        elif rand==2:

            game = MiniGame\_Two()

        elif rand==3:

            game = flappyBird()

        else:

             ("HGBI\*YGBI\*YB")

        game.Play\_Mini()

        if game.getWon():

            player.coins+=1

            return player

        player.move(-3)

        return player

    def red\_land\_2():

        temp\_coins=player.coins

        player.lose\_coins(4)

        player.coins=temp\_coins

        return player

    def red\_land\_3():

        rand=random.randrange(1,4)

        if rand==1:

            game = MiniGame\_One()

        elif rand==2:

            game = MiniGame\_Two()

        elif rand==3:

            game = flappyBird()

        else:

             ("HGBI\*YGBI\*YB")

        game.Play\_Mini()

        if game.getWon():

            player.coins+=1

            return player

        player.lose\_coins(5)

        return player

    if rand==1:

        t = InputText("move back minigame")

        return red\_land\_1()

    elif rand==2:

        t = InputText("purse check minigame")

        return red\_land\_2()

    elif rand==3:

        t = InputText("lose coins minigame")

        return red\_land\_3()

    else:

        return player

def green\_land(player):

    rand=random.randrange(1,4)

    rand=3

    def green\_land\_1():

        rand=random.randrange(1,4)

        player.coins+=rand

        t = InputText("you gained "+str(rand)+" coins")

        return player

    def green\_land\_2():

        rand=random.randrange(1,4)

        if rand==1:

            game = MiniGame\_One()

        elif rand==2:

            game = MiniGame\_Two()

        elif rand==3:

            game = flappyBird()

        else:

             ("HGBI\*YGBI\*YB")

        game.Play\_Mini()

        if game.getWon():

            player.coins+=4

            return player

        player.lose\_coins(1)

        return player

    def green\_land\_3():

        t = InputText("welcome to the shop you have  "+str(player.coins)+" coins")

        t = InputText("enter 1 to move forward 5 spaces for 3 coins or enter 2 to add 1 to your rolls for 4 coins, enter 3 to pruchase neither")

        aceptable=["1","2","3"]

        if t.getInput() in aceptable:

            x=t.getInput()

            if x=="1":

                if player.coins<3:

                    t = InputText("insufisent funds you only have  "+str(player.coins)+" coins")

                    return player

                player.lose\_coins(3)

                player.move(5)

            elif x=="2":

                if player.coins<4:

                    t = InputText("insufisent funds you only have  "+str(player.coins)+" coins")

                    return player

                player.lose\_coins(4)

                player.min\_roll+=1

                player.max\_roll+=1

        else:

            i= InputText("invalid input "+t.getInput())

        return player

    if rand==1:

        return green\_land\_1()

    elif rand==2:

        t = InputText("money gain minigame")

        return green\_land\_2()

    elif rand==3:

        t = InputText("shop")

        return green\_land\_3()

    else:

        return player

def play(menu,players):

    global root

    global  board

    root = Tk()

    root.title("Board")

    board = Board(menu,players)

    root.winfo\_toplevel().positionfrom("user")

    root.mainloop()

import time

import pygame

import sys

# Set the timer duration in seconds

class Timer():

    def \_\_init\_\_(self,timer\_duration,screen,size):

        self.timer\_duration=timer\_duration

        self.screen=screen

        self.start\_time = time.time()

        self.size=size

    def get\_font(size): # Returns Press-Start-2P in the desired size

        return pygame.font.Font("images\Font.ttf", size)

    def run\_timer(self):

        pygame.font.init()

        while True:

            elapsed\_time = time.time() -self.start\_time

            remaining\_time = self.timer\_duration - elapsed\_time

            TIME\_MSG = Timer.get\_font(self.size).render(f"Time remaining: {remaining\_time:.2f} seconds", True, "green")

            TIME\_RECT=TIME\_MSG.get\_rect(center=(650,10))

            self.screen.blit(TIME\_MSG,TIME\_RECT)

            if remaining\_time <= 0:

                return True

            return False

    def win\_game(self):

        pygame.font.init()

        while True:

            self.screen.fill("white")

            WIN\_TEXT=Timer.get\_font(40).render("You have won",True,"Blue")

            WIN\_RECT=WIN\_TEXT.get\_rect(center=(400, 260))

            self.screen.blit(WIN\_TEXT,WIN\_RECT)

            WIN\_TEXTTWO =Timer.get\_font(20).render("Click the window to continue", True, "Blue")

            WIN\_RECTTWO=WIN\_TEXT.get\_rect(center=(450, 360))

            self.screen.blit(WIN\_TEXTTWO,WIN\_RECTTWO)

            for event in pygame.event.get():

                if event.type==pygame.QUIT:

                    pygame.quit()

                    sys.exit()

                if event.type == pygame.MOUSEBUTTONDOWN:

                    pygame.quit()

                    return True

            #time.sleep(1)

            pygame.display.update()

    def lost\_game(self):

        pygame.font.init()

        while True:

            self.screen.fill("white")

            LOST\_TEXT=Timer.get\_font(40).render("You lost the game ",True,"Blue")

            LOST\_RECT=LOST\_TEXT.get\_rect(center=(400, 260))

            self.screen.blit(LOST\_TEXT,LOST\_RECT)

            LOST\_TEXTTWO =Timer.get\_font(20).render("Click the window to continue", True, "Blue")

            LOST\_RECTTWO=LOST\_TEXT.get\_rect(center=(450, 360))

            self.screen.blit(LOST\_TEXTTWO,LOST\_RECTTWO)

            for event in pygame.event.get():

                if event.type==pygame.QUIT:

                    pygame.quit()

                    sys.exit()

                if event.type == pygame.MOUSEBUTTONDOWN:

                    pygame.quit()

                    return True

            pygame.display.update()

import pygame

import sys

class InputText:

    def \_\_init\_\_(self,prompt="prompt"):

        pygame.init()

        self.clock = pygame.time.Clock()

        self.screen = pygame.display.set\_mode([1000, 400])

        self.base\_font = pygame.font.Font(None, 24)

        self.fix\_text = prompt

        self.user\_text  = ""

        self.text\_lines = []  # Store lines of text

        self.leftConer\_x = 400

        self.leftConer\_y = 10

        self.color = pygame.Color( (255, 255, 255))

        self.rect = pygame.Rect(self.leftConer\_x,self.leftConer\_y,380,40)

        self.runInputBox()

    '''

    def getInput(self):

        return '\n'.join(self.text\_lines)

    '''

    def getInput(self):

        return self.user\_text

    def runInputBox(self):

        while True:

            for event in pygame.event.get():

                if event.type == pygame.QUIT:

                    pygame.quit()

                    return

                if event.type == pygame.KEYDOWN:

                    if event.key == pygame.K\_RETURN:  # Check for Enter key

                        #self.text\_lines.append(self.user\_text)

                        input\_text = self.getInput()  # Retrieve the text input

                        print(input\_text)

                        pygame.quit()

                        return

                        #self.user\_text = ""  # Reset for new line

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

                        self.user\_text = self.user\_text[:-1]  # Remove last character

                    else:

                        self.user\_text += event.unicode

            self.screen.fill((0, 0, 0))

            #text\_width = max(350, self.base\_font.size(self.user\_text)[0] + 10)

            #self.rect.w = text\_width

            #pygame.draw.rect(self.screen,self.color,self.rect,2)

            text\_surfaceFixed = self.base\_font.render(self.fix\_text, True, (255, 255, 255))

            self.screen.blit(text\_surfaceFixed, (0, self.leftConer\_y+10))

            # Render existing lines

            y\_offset = 20

            for line in self.text\_lines:

                text\_surface = self.base\_font.render(line, True, (255, 255, 255))

                 # Adjust vertical offset

                y\_offset += text\_surface.get\_height()

                self.screen.blit(text\_surface, (170,y\_offset))

            # Render the current line being typed

            current\_text\_surface = self.base\_font.render(self.user\_text, True, (255, 255, 255))

            self.screen.blit(current\_text\_surface,(20,50)) #( self.rect.x + 5, self.rect.y + 5))

            if current\_text\_surface.get\_width() >= 600:

                self.text\_lines.append(self.user\_text)

                self.user\_text = ""

            pygame.display.flip()

            self.clock.tick(60)

# Import the required libraries

import pygame

import random

from pygame.locals import Rect

import math

from Timer import Timer  # Import Timer class from Timer module

import sys

# Initialize the game engine

pygame.init()

BLACK = [0, 0, 0]

WHITE = [255, 255, 255]

class MiniGame\_One:

    def \_\_init\_\_(self):

        # Create an empty array

        self.window\_width = 800

        self.window\_height = 800

        # Set the height and wi dth of the screen

        SIZE = [self.window\_width,self.window\_height]

        self.screen = pygame.display.set\_mode(SIZE)

        pygame.display.set\_caption("Snow Animation")

        self.snow\_list = []

        self.spaceship = pygame.image.load("images\spaceship.png")

        self.spaceship = pygame.transform.scale(self.spaceship, (40, 40))

        self.spaceship\_x = self.window\_width // 2

        self.spaceship\_y = self.window\_height - 50

        self.spaceship\_rect = Rect(self.spaceship\_x, self.spaceship\_y, 40, 40)

        self.collided\_snowflakes = []

        self.frame\_count = 1

        self.generator()

        self.clock = pygame.time.Clock()

        self.count = 0

        self.clockTime = 500

        self.won = False

        self.lose = False

    def distance(self, point1, point2):

        return math.sqrt((point1[0] - point2[0])\*\*2 + (point1[1] - point2[1])\*\*2)

    def generator(self):

        if self.frame\_count % 200 == 0:

            for i in range(5):

                x = random.randrange(0, 800)

                y = random.randrange(0, 100)

                self.snow\_list.append([x, y])

    def getLost(self):

        return self.lose

    def getWon(self):

        return self.won

    def Play\_Mini(self):

        done = False

        timer = Timer(10, self.screen, 10)

        while not done:

            for event in pygame.event.get():

                if event.type == pygame.QUIT:

                    done = True

                    pygame.quit()

                    sys.exit()

            self.screen.fill(BLACK)

            if timer.run\_timer():

                self.won=timer.win\_game()

                print(self.won)

                break

            keys = pygame.key.get\_pressed()

            if keys[pygame.K\_LEFT]:

                self.spaceship\_x -= 1

            if keys[pygame.K\_RIGHT]:

                self.spaceship\_x += 1

            if keys[pygame.K\_UP]:

                self.spaceship\_y -= 1

            if keys[pygame.K\_DOWN]:

                self.spaceship\_y += 1

            self.screen.blit(self.spaceship, (self.spaceship\_x, self.spaceship\_y))

            self.generator()

            self.frame\_count += 1

            for i in range(len(self.snow\_list)):

                snowflake\_rect=pygame.draw.circle(self.screen, WHITE, self.snow\_list[i], 2)

                snowflake\_x, snowflake\_y = self.snow\_list[i]

                snowflake\_center = [snowflake\_x, snowflake\_y]

                if self.distance(snowflake\_center, [self.spaceship\_x + 20, self.spaceship\_y + 20]) <= 20:

                    if i not in self.collided\_snowflakes:

                        self.collided\_snowflakes.append(i)

                        self.count += 1

                        #print("Collision detected " + str(self.count))

                        self.lose=timer.lost\_game()

                        #print(self.won)

                        return

                        break

                self.snow\_list[i][1] += 1

            pygame.display.flip()

            self.clock.tick(500)

        pygame.quit()

import pygame

import random

from Timer import Timer

import sys

pygame.init()

WIDTH, HEIGHT = 800, 800

size = (WIDTH, HEIGHT)

WHITE = (255, 255, 255)

BLACK = (0, 0, 0)

RED = (255, 0, 0)

class MiniGame\_Two:

    def \_\_init\_\_(self):

        self.screen = pygame.display.set\_mode(size)

        pygame.display.set\_caption("Board")

        self.rect1\_x, self.rect1\_y = 100, 200  # Initial position for the first rectangle

        self.rect2\_x, self.rect2\_y = 700, 400  # Initial position for the second rectangle

        self.x\_start, self.y\_start = 100, 770

        self.x\_end, self.y\_end = 300, 770

        self.start\_point = (self.x\_start, self.y\_start)

        self.end\_point = (self.x\_end, self.y\_end)

        self.clock = pygame.time.Clock()

        self.rect1\_change\_x, self.rect1\_change\_y = 3, 3

        self.rect2\_change\_x, self.rect2\_change\_y = 3, 3

        self.done = False

        self.timer = Timer(10, self.screen, 10)

        self.won = False

        self.lose = False

    def timerStart(self):

        timer = Timer(10, self.screen, 10)

        return timer

    def getLost(self):

        return self.lose

    def getWon(self):

        return self.won

    def Play\_Mini(self):

        while not self.done:

            for event in pygame.event.get():

                if event.type == pygame.QUIT:

                    self.done = True

                    pygame.quit()

                    sys.exit()

            self.screen.fill(BLACK)

            rect = pygame.draw.rect(self.screen, WHITE, [self.rect2\_x, self.rect2\_y, 50, 50])

            rec2 = pygame.draw.rect(self.screen, WHITE, [self.rect1\_x, self.rect1\_y, 50, 50])

            pygame.draw.rect(self.screen, RED, [self.rect1\_x + 10, self.rect1\_y + 10, 30, 30])

            if self.timer.run\_timer():

                self.won=self.timer.win\_game()

                break

            keys = pygame.key.get\_pressed()

            if keys[pygame.K\_LEFT]:

                self.x\_start -= 10

                self.x\_end -= 10

            if keys[pygame.K\_RIGHT]:

                self.x\_start += 10

                self.x\_end += 10

            start\_point = (self.x\_start, self.y\_start)

            end\_point = (self.x\_end, self.y\_end)

            line = pygame.draw.line(self.screen, WHITE, start\_point, end\_point, 5)

            self.rect1\_x += self.rect1\_change\_x

            self.rect1\_y += self.rect1\_change\_y

            self.rect2\_x += self.rect2\_change\_x

            self.rect2\_y += self.rect2\_change\_y

            if line.colliderect(self.rect2\_x, self.rect2\_y, 50, 50):

                while -4<self.rect2\_change\_x and self.rect2\_change\_x<4:

                    self.rect2\_change\_x = random.randint(-5, 5)

                self.rect2\_change\_y \*= -1

            if line.colliderect(self.rect1\_x, self.rect1\_y, 50, 50):

                while -4<self.rect1\_change\_x and self.rect1\_change\_x<4:

                    self.rect1\_change\_x = random.randint(-5, 5)

                self.rect1\_change\_y \*= -1

            if rect.colliderect(self.rect1\_x, self.rect1\_y, 50, 50):

                while -4<self.rect1\_change\_x and self.rect1\_change\_x<4:

                    self.rect1\_change\_x = random.randint(-5, 5)

                self.rect1\_change\_y \*= -1

            if rec2.colliderect(self.rect2\_x, self.rect2\_y, 50, 50):

                while -4<self.rect2\_change\_x and self.rect2\_change\_x<4:

                    self.rect2\_change\_x = random.randint(-5, 5)

                self.rect2\_change\_y \*= -1

            if self.rect1\_y > 750:

                self.lose=self.timer.lost\_game()

                return

                break

            if self.rect2\_y > 750:

                self.lose=self.timer.lost\_game()

                return

                break

            if self.rect1\_y > 750 or self.rect1\_y < 0:

                self.rect1\_change\_y \*= -1

            if self.rect1\_x > 750 or self.rect1\_x < 0:

                self.rect1\_change\_x \*= -1

            if self.rect2\_y > 750 or self.rect2\_y < 0:

                self.rect2\_change\_y \*= -1

            if self.rect2\_x > 750 or self.rect2\_x < 0:

                self.rect2\_change\_x \*= -1

            pygame.display.flip()

            self.clock.tick(60)

import pygame

import sys

import random

from pygame.locals import \*

from Timer import \*

# Set up display

WIDTH, HEIGHT = 800, 800

WHITE = (255, 255, 255)

BLACK = (0, 0, 0)

BLUE = (0, 0, 255)

class  flappyBird:

    def \_\_init\_\_(self):

        pygame.init()

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

        pygame.display.set\_caption("Flappy Bird")

        # Set up bird

        self.bird\_size = 30

        self.bird\_x = 500

        self.bird\_y = HEIGHT // 4 - self.bird\_size // 2

        self.bird\_speed = 5

        # Set up pipes

        self.pipe\_width = 50

        self.pipe\_gap = 400  # Increased pipe gap

        self.pipe\_speed = 3

        self.pipes = []

        # Set up game variables

        self.gravity = 0.9  # Reduced gravity

        self.jump\_strength = 12

        self.bird\_alive = True

        self.timer = Timer(10, self.screen, 10)

        self.won = False

        self.lose = False

    def getLost(self):

        return self.lose

    def getWon(self):

        return self.won

    def timerStart(self):

        timer = Timer(10, self.screen, 10)

        return timer

    # Function to draw the bird

    def draw\_bird(self):

        pygame.draw.rect(self.screen, BLUE, (self.bird\_x, self.bird\_y, self.bird\_size, self.bird\_size))

    # Function to draw the pipes

    def draw\_pipes(self):

        for pipe\_pair in self.pipes:

            pygame.draw.rect(self.screen, BLACK, (pipe\_pair[0], 0, self.pipe\_width, pipe\_pair[1]))

            pygame.draw.rect(self.screen, BLACK, (pipe\_pair[0], pipe\_pair[1] + self.pipe\_gap, self.pipe\_width, HEIGHT - pipe\_pair[1] - self.pipe\_gap))

    # Function to move the bird

    def move\_bird(self):

        self.bird\_y += self.bird\_speed

        self.bird\_speed += self.gravity

    # Function to check for collisions with pipes

    def check\_collision(self):

        global bird\_alive

        if self.bird\_y < 0 or self.bird\_y + self.bird\_size > HEIGHT:

            bird\_alive = False

            self.lose=self.timer.lost\_game()

            return self.lose

        for pipe\_pair in self.pipes:

            if (

                self.bird\_x < pipe\_pair[0] + self.pipe\_width

                and self.bird\_x + self.bird\_size > pipe\_pair[0]

                and (self.bird\_y < pipe\_pair[1] or self.bird\_y + self.bird\_size > pipe\_pair[1] + self.pipe\_gap)

            ):

                bird\_alive = False

                self.lose=self.timer.lost\_game()

                return self.lose

    def Play\_Mini(self):

        # Main game loop

        while True:

            # Event handling

            for event in pygame.event.get():

                if event.type == QUIT:

                    pygame.quit()

                    sys.exit()

                elif event.type == KEYDOWN:

                    if event.key == K\_SPACE and self.bird\_alive:

                        self.bird\_speed = -self.jump\_strength

            # Move bird

            if self.bird\_alive:

                self.move\_bird()

            # Move pipes

            for self.pipe\_pair in self.pipes:

                self.pipe\_pair[0] -= self.pipe\_speed

            # Spawn new pipes

            if not self.pipes or self.pipes[-1][0] < WIDTH - 200:

                self.pipe\_height = random.randint(50, HEIGHT - self.pipe\_gap - 50)

                self.pipes.append([WIDTH, self.pipe\_height])

            # Remove off-screen pipes

            self.pipes = [self.pipe\_pair for self.pipe\_pair in self.pipes if self.pipe\_pair[0] + self.pipe\_width > 0]

            # Check for collisions

            if(self.check\_collision()):

                return

            # Clear the screen

            self.screen.fill(WHITE)

            # Draw bird and pipes

            self.draw\_bird()

            self.draw\_pipes()

            if self.timer.run\_timer():

                self.won=self.timer.win\_game()

                return

            # Update display

            pygame.display.update()

            # Set the frame rate

            pygame.time.Clock().tick(30)

import pygame

import random

from Timer import \*

# Initialize Pygame

pygame.init()

# Constants

WIDTH, HEIGHT = 800, 800

GRID\_SIZE = 20

FPS = 10

# Colors

WHITE = (255, 255, 255)

RED = (255, 0, 0)

GREEN = (0, 255, 0)

BLACK = (0, 0, 0)

class snake:

# Create the game window

    def \_\_init\_\_(self):

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

        pygame.display.set\_caption("Snake Game")

        # Snake and food variables

        self.snake = [(200, 200)]

        self.snake\_dir = "RIGHT"

        self.snake\_speed = 5

        self.food = (random.randint(0, WIDTH - GRID\_SIZE) // GRID\_SIZE \* GRID\_SIZE,

                    random.randint(0, HEIGHT - GRID\_SIZE) // GRID\_SIZE \* GRID\_SIZE)

        self.foodTwo = (random.randint(0, WIDTH - GRID\_SIZE-50) // GRID\_SIZE \* GRID\_SIZE,

                    random.randint(0, HEIGHT - GRID\_SIZE-50) // GRID\_SIZE \* GRID\_SIZE)

        self.foodThree = (random.randint(0, WIDTH - GRID\_SIZE-50) // GRID\_SIZE \* GRID\_SIZE,

                    random.randint(0, HEIGHT - GRID\_SIZE-50) // GRID\_SIZE \* GRID\_SIZE)

        self.clock = pygame.time.Clock()

        self.timer = Timer(60, self.screen, 10)

        self.won = False

        self.lose = False

    def getLost(self):

        return self.lose

    def getWon(self):

        return self.won

    def timerStart(self):

        timer = Timer(10, self.screen, 10)

        return timer

# Game over condition

    def game\_over(self):

        self.font = pygame.font.Font(None, 36)

        self.text = self.font.render("Game Over! Press Q to quit or C to play again", True, RED)

        self.text\_rect = self.text.get\_rect(center=(WIDTH // 2, HEIGHT // 2))

        self.screen.blit(self.text, self.text\_rect)

        pygame.display.update()

        self.game\_over\_flag = True

        while self.game\_over\_flag:

            for event in pygame.event.get():

                if event.type == pygame.QUIT:

                    pygame.quit()

                    quit()

                if event.type == pygame.KEYDOWN:

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

                        pygame.quit()

                        quit()

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

                        self.reset\_game()

                        self.run\_game()

# Function to reset the game

    def reset\_game(self):

        self.snake, self.snake\_dir, self.snake\_speed, self.food

        self.snake = [(random.randint(0, WIDTH - GRID\_SIZE) // GRID\_SIZE \* GRID\_SIZE,

                    random.randint(0, HEIGHT - GRID\_SIZE) // GRID\_SIZE \* GRID\_SIZE)]

        self.snake\_dir = "RIGHT"

        self.snake\_speed = 20

        self.food = (random.randint(0, WIDTH - GRID\_SIZE) // GRID\_SIZE \* GRID\_SIZE,

                    random.randint(0, HEIGHT - GRID\_SIZE) // GRID\_SIZE \* GRID\_SIZE)

# Function for the main game loop

    def Play\_Mini(self):

        self.snake, self.snake\_dir, self.food,self.foodTwo,self.foodThree, self.snake\_speed

        self.game\_exit = False

        while not self.game\_exit:

            for event in pygame.event.get():

                if event.type == pygame.QUIT:

                    self.game\_exit = True

                elif event.type == pygame.KEYDOWN:

                    #a,w,s,d

                    if event.key == pygame.K\_w and self.snake\_dir != "DOWN":

                        self.snake\_dir = "UP"

                    elif event.key == pygame.K\_s and self.snake\_dir != "UP":

                        self.snake\_dir = "DOWN"

                    elif event.key == pygame.K\_a and self.snake\_dir != "RIGHT":

                        self.snake\_dir = "LEFT"

                    elif event.key == pygame.K\_d and self.snake\_dir != "LEFT":

                        self.snake\_dir = "RIGHT"

                    #arrows

                    if event.key == pygame.K\_UP and self.snake\_dir != "DOWN":

                        self.snake\_dir = "UP"

                    elif event.key == pygame.K\_DOWN and self.snake\_dir != "UP":

                        self.snake\_dir = "DOWN"

                    elif event.key == pygame.K\_LEFT and self.snake\_dir != "RIGHT":

                        self.snake\_dir = "LEFT"

                    elif event.key == pygame.K\_RIGHT and self.snake\_dir != "LEFT":

                        self.snake\_dir = "RIGHT"

        # Move the snake

            if self.snake\_dir == "UP":

                self.new\_head = (self.snake[0][0], self.snake[0][1] - GRID\_SIZE)

            elif self.snake\_dir == "DOWN":

                self.new\_head = (self.snake[0][0], self.snake[0][1] + GRID\_SIZE)

            elif self.snake\_dir == "LEFT":

                self.new\_head = (self.snake[0][0] - GRID\_SIZE, self.snake[0][1])

            elif self.snake\_dir == "RIGHT":

                self.new\_head = (self.snake[0][0] + GRID\_SIZE, self.snake[0][1])

            self.snake.insert(0, self.new\_head)

        # Check for collision with food

            if self.new\_head == self.food or self.new\_head == self.foodTwo or self.new\_head == self.foodThree :

                self.food = (random.randint(0, WIDTH - GRID\_SIZE) // GRID\_SIZE \* GRID\_SIZE,

                        random.randint(0, HEIGHT - GRID\_SIZE) // GRID\_SIZE \* GRID\_SIZE)

                self.foodTwo = (random.randint(0, WIDTH - GRID\_SIZE) // GRID\_SIZE \* GRID\_SIZE,

                        random.randint(0, HEIGHT - GRID\_SIZE) // GRID\_SIZE \* GRID\_SIZE)

                self.foodThree = (random.randint(0, WIDTH - GRID\_SIZE) // GRID\_SIZE \* GRID\_SIZE,

                        random.randint(0, HEIGHT - GRID\_SIZE) // GRID\_SIZE \* GRID\_SIZE)

                self.snake\_speed += 1

            #else:

                #self.snake.pop()

        # Check for collision with boundaries or itself

            if (self.new\_head[0] >= WIDTH or self.new\_head[0] < 0 or

                    self.new\_head[1] >= HEIGHT or self.new\_head[1] < 0 or

                    self.new\_head in self.snake[1:]):

                self.lose=self.timer.lost\_game()

                return

        # Draw everything

            self.screen.fill(BLACK)

            pygame.draw.rect(self.screen, GREEN, (\*self.food, GRID\_SIZE, GRID\_SIZE))

            pygame.draw.rect(self.screen, RED, (\*self.foodTwo, GRID\_SIZE, GRID\_SIZE))

            pygame.draw.rect(self.screen, RED, (\*self.foodThree, GRID\_SIZE, GRID\_SIZE))

            for segment in self.snake:

                pygame.draw.rect(self.screen, WHITE, (\*segment, GRID\_SIZE, GRID\_SIZE))

            if self.timer.run\_timer():

                self.won=self.timer.win\_game()

                break

            pygame.display.update()

            self.clock.tick(self.snake\_speed)

        pygame.quit()