**import** random

**import** pygame

**import** os

**import** time

**import** pickle

**import** socket

**from** tkinter **import** \*

**from** tkinter **import** messagebox as ms

**import** sqlite3

**from** sys **import** exit

**from** networkclass **import** Network

**from** player **import** MOplayer

**from** \_thread **import** \*

#imported modules necessary for game's function

pygame.mixer.pre\_init(44100,16,2,4096) #initialises pygame mixer for music

pygame.init() #initialises pygame

colouroption = False

Red = False

Blue = False

Green = False

Yellow = False

skipturn = False #variable used to check if a skip action card has been played

reverseturn = False #variable used to check if a skip reverse card has been played

p1reverseturn = False

p1skipturn = False

p2reverseturn = False

p2skipturn = False

Playerwins = False  #variable used to check if Player has won

Computerwins = False   #variable used to check if Computer has won

Player1wins = False #variable used to chegck if Player1 in multiplayer has won

Player2wins = False #variable used to check if Player2 in multiplayer has won

Score = 0 #placeholder for score value for users (yet to be added t table)

x = 800 #screen resolution width

y = 600 #screen resolution height

div\_iwidth = 750 #adjustment to width for dispalying images

div\_iheight = 550 #adjustment to height for dispalying images

white = (255,255,255)

yellow = (255,255,0)

black = (0,0,0)

green = (0,255,0)

red = (255,0,0)

blue = (0,0,255)

orange = (255,165,0)  #predefined colours

mouseposition = pygame.mouse.get\_pos() #gets mouse position

**class** Card:

**def** \_\_init\_\_(self, suit, number): #Card given the attribute Suit and Number

        self.suit = suit

        vals = {10: "reverse", 11:"skip", 12:"+2", 13:"4x", 14:"0"} #dictionary to store action card values paired with number keys

**if** type(number) **is** int:

**if** number < 10:

                self.number = str(number)

**else**:

                self.number = vals[number] #used when building the deck of cards, to create action cards using above dictionary

        self.image = pygame.image.load(str(self) + ".png") #load each creating image into the program according to the card name

**def** \_\_eq\_\_(self, other):

**if** self.suit == other.suit **and** self.number == other.number:

**return** True

**return** False

**def** \_\_repr\_\_(self):

**return** self.showcard()

**def** getImage():

**return** self.image

**def** showcard(self):

**return** "{} {}".format(self.suit, self.number) #returns value of a created Card in a printed format

**class** Deck:

**def** \_\_init\_\_(self):

        self.cards = [] #an array stored all the card values

        self.builddeck()

**def** builddeck(self):

**for** s **in** ["yellow","red","blue","green"]:

**for** n **in** range(0,13):

                self.cards.append(Card(s, n))   #builds a deck by pairing each colour with numbers from 0 to 12, then converting them into a Card object and adding them to an array of card

**for** i **in** range (13,15):

            self.cards.append(Card("wild", i))

**def** showdeck(self):

**for** card **in** self.cards:

**print**(card)  #when showdeck is called, it will display all cards in the self.cards array

**def** shuffle(self):

        random.shuffle(self.cards) #uses random to shuffle order of cards array

**def** drawcard(self):

**return** self.cards.pop()  #removes last card in cards array and returns the value

**class** Player:

**def** \_\_init\_\_(self, name):

        self.hand = [] #array for player's hand

        self.name = name

**def** draw(self, deck, x):

**for** i **in** range(x):

            self.hand.append(deck.drawcard())

            #allows player draw multiple cards and appends them into hand array

**def** showhand(self):

**global** Score

**print**("{} Hand is: ".format(self.name))

      h=30

**for** card **in** self.hand:

**print**(card)

          maingame.displayimage(card.image,div\_iwidth,div\_iheight-h) #display's each card in a player's hand on to the game screen

          pygame.display.update()

          h=h+25 #moves each following card down by 25 to ensure the user can see all the cards

**if** len(self.hand) == 0:

        Playerwins = True

        maingame.add\_screen()

        maingame.gametext\_display("Player1 won. Game Over", 2, 2, 40)

        maingame.gametext\_display("Score: +100 ", 2,4,40)

        Score = Score + 100

**print**("Youre final score is ",Score)

        maingame.updatescore(Score, unodatabase.username)

        pygame.mixer.music.load("winnermusic.mp3")

        pygame.mixer.music.play(-1)

        pygame.display.update()

        time.sleep(5)

        pygame.quit()

        os.\_exit(1)

        #once a player has no cards in their hand, Player has won, displays win screen and plays winner music

**elif** len(self.hand) == 1:

**global** t

**global** unoconfirmed

        unoconfirmed = False

        t = 1

        unoroot = Tk()

        Button(unoroot, text="Click to call uno", command=maingame.unocalled).pack()

        unoroot.mainloop()

**while** t **and** unoconfirmed == False:

            mins, secs = divmod(t, 60)

            timer = '{:02d}:{:02d}'.format(mins, secs)

**print**(timer, end="\r")

            time.sleep(1)

            t -= 1

**if** t == 0 **and** unoconfirmed == False:

            ms.showerror("","Time out! Penalty: draw a card")

            Player1.draw(deck,1)

            Player.showhand()

            Computer.showhand()

**elif** t == 0 **and** unoconfirmed == True:

            ms.showinfo("","UNO Called")

            time.sleep(1)

            #continue

        #timer aspect not working but uno function is there

**def** discard(self):

**global** skipturn

**global** reverseturn

**if** len(self.hand) != 0:

        discard\_card = self.hand[maingame.down]

        self.throwAway(discard\_card) #if the player has cards left in their hand, it will discard the card based on their keyboard input using throwAway

**print**(discard\_card.suit)

**def** throwAway(self, discard):

**global** Score

**global** maingamepile

**if** len(maingamepile) > 1: #checks if there is a card or cards in play already

        lastcardplaced = maingamepile[-1] #if there is then the last card on the pile = the last card placed

**for** card **in** self.hand:

**if** card == discard:

**if** card.number == lastcardplaced.number **or** card.suit == lastcardplaced.suit: #checks if card selected to be discarded has the same suit or number as the last card in play on the pile

                    self.hand.remove(card) #card is removed from player's hand

                    maingamepile.append(card) #card is added on to main game pile

                    maingame.gametext\_display("Player1 played this card", 2,5, 20)

                    time.sleep(1)

                    maingame.display\_last\_discarded()

**break**

**elif** card.number == "+2":

                        maingame.gametext\_display("Computer Draws 2 more cards", 2, 4,20)

                        time.sleep(1)

                        #pygame.display.update()

                        Computer.draw(deck, 2) #if card is a +2, computer gets 2 more cards

                        self.hand.remove(card) #card is removed from player's hand

                        maingamepile.append(card) #card is added on to main game pile

                        Score = Score + 50

                        maingame.gametext\_display("Player1 played this card", 2,5, 20)

                        time.sleep(1)

                        maingame.display\_last\_discarded()

**elif** card.number == "skip":

                        skipturn = True

                        maingame.gametext\_display("Computer's Turn Will Be Skipped Next Round", 2, 4,20)

                        time.sleep(1)

                        #pygame.display.update() #if card is a skip, computer's turn will be skipped next in the main game loop

                        self.hand.remove(card) #card is removed from player's hand

                        maingamepile.append(card) #card is added on to main game pile

                        Score = Score + 50

                        maingame.gametext\_display("Player1 played this card", 2,5, 20)

                        time.sleep(1)

                        maingame.display\_last\_discarded()

**elif** card.number == "reverse":

                        reverseturn = True

                        maingame.gametext\_display("Computer's Turn Will Be Reversed Next Round", 2,4,20)

                        time.sleep(1)

                        #pygame.display.update() #if card is a reverse, computer's turn will be reversed next in the main game loop (effectively player gets another free turn)

                        self.hand.remove(card) #card is removed from player's hand

                        maingamepile.append(card) #card is added on to main game pile

                        Score = Score + 50

                        maingame.gametext\_display("Player1 played this card", 2,5, 20)

                        time.sleep(1)

                        maingame.display\_last\_discarded()

**elif** card.suit == "wild":

**print**("wild card played")

                    Score = Score + 70

                    maingame.gametext\_display("Computer draws 4 cards", 2, 4, 20)

                    time.sleep(1)

                    #pygame.display.update()

                    Computer.draw(deck, 4)

                    maingame.colourchangescreen()

**if** Red == True:

**for** card **in** self.hand:

**if** card.suit == "red":

                                self.hand.remove(card)

                                maingamepile.append(card)

                                maingame.gametext\_display("Player1 played this card", 2,5, 15)

                                maingame.display\_last\_discarded()

**else**:

**print**("no red card")

**continue**

**elif** Green == True:

**for** card **in** self.hand:

**if** card.suit == "green":

                                self.hand.remove(card)

                                maingamepile.append(card)

                                maingame.gametext\_display("Player1 played this card", 2,5, 15)

                                maingame.display\_last\_discarded()

**else**:

**print**("no green card")

**continue**

**elif** Blue == True:

**for** card **in** self.hand:

**if** card.suit == "blue":

                                self.hand.remove(card)

                                maingamepile.append(card)

                                maingame.gametext\_display("Player1 played this card", 2,5, 15)

                                maingame.display\_last\_discarded()

**else**:

**print**("no blue card")

**continue**

**elif** Yellow == True:

**for** card **in** self.hand:

**if** card.suit == "yellow":

                                self.hand.remove(card)

                                maingamepile.append(card)

                                maingame.gametext\_display("Player1 played this card", 2,5, 15)

                                maingame.display\_last\_discarded()

**else**:

**print**("no yellow card")

**continue**

**else**:

                    invalidturn = True

                    maingame.gametext\_display("Invalid move. 1 card added to hand.",2,5,15)

                    #pygame.display.update()

                    Player1.draw(deck, 1)

**break**  #if the user selects a card that isn't the same suit or number the card is invalidly played and they will face a draw card penalty

**else**:

**for** card **in** self.hand:

**if** card == discard:

            self.hand.remove(card)

            maingamepile.append(card) #if there are no cards in play on the main game pile then the card the user selects will be discarded as normal (as this means the user is starting the game first)

            maingame.display\_last\_discarded()

**class** Playerone(Player):

**def** throwAway(self, discard):

**global** Score

**global** maingamepile

**if** len(maingamepile) > 1: #checks if there is a card or cards in play already

        lastcardplaced = maingamepile[-1] #if there is then the last card on the pile = the last card placed

**for** card **in** self.hand:

**if** card == discard:

**if** card.number == lastcardplaced.number **or** card.suit == lastcardplaced.suit: #checks if card selected to be discarded has the same suit or number as the last card in play on the pile

                    self.hand.remove(card) #card is removed from player's hand

                    maingamepile.append(card) #card is added on to main game pile

**break**

**elif** card.number == "+2":

                        maingame.gametext\_display("Player2 Draws 2 more cards", 2, 4,15)

                        pygame.display.update()

                        mPlayer2.draw(deck, 2) #if card is a +2, computer gets 2 more cards

                        self.hand.remove(card) #card is removed from player's hand

                        maingamepile.append(card) #card is added on to main game pile

**elif** card.number == "skip":

                        p1skipturn = True

                        maingame.gametext\_display("Player2's Turn Will Be Skipped Next Round", 2, 4,15)

                        pygame.display.update() #if card is a skip, computer's turn will be skipped next in the main game loop

                        self.hand.remove(card) #card is removed from player's hand

                        maingamepile.append(card) #card is added on to main game pile

**elif** card.number == "reverse":

                        p1reverseturn = True

                        maingame.gametext\_display("Player2's Turn Will Be Reversed Next Round", 2,4,15)

                        pygame.display.update() #if card is a reverse, computer's turn will be reversed next in the main game loop (effectively player gets another free turn)

                        self.hand.remove(card) #card is removed from player's hand

                        maingamepile.append(card) #card is added on to main game pile

**elif** card.suit == "wild":

**print**("wild card played")

                    Score = Score + 70

                    maingame.gametext\_display("Player2 draws 4 cards", 2, 4, 15)

                    #pygame.display.update()

                    mPlayer2.draw(deck, 4)

                    maingame.colourchangescreen()

**if** Red == True:

**for** card **in** self.hand:

**if** card.suit == "red":

                                self.hand.remove(card)

                                maingamepile.append(card)

                                maingame.gametext\_display("Player1 played this card", 2,5, 15)

                                maingame.display\_last\_discarded()

**else**:

**print**("no red card")

**continue**

**elif** Green == True:

**for** card **in** self.hand:

**if** card.suit == "green":

                                self.hand.remove(card)

                                maingamepile.append(card)

                                maingame.gametext\_display("Player1 played this card", 2,5, 15)

                                maingame.display\_last\_discarded()

**else**:

**print**("no green card")

**continue**

**elif** Blue == True:

**for** card **in** self.hand:

**if** card.suit == "blue":

                                self.hand.remove(card)

                                maingamepile.append(card)

                                maingame.gametext\_display("Player1 played this card", 2,5, 15)

                                maingame.display\_last\_discarded()

**else**:

**print**("no blue card")

**continue**

**elif** Yellow == True:

**for** card **in** self.hand:

**if** card.suit == "yellow":

                                self.hand.remove(card)

                                maingamepile.append(card)

                                maingame.gametext\_display("Player1 played this card", 2,5, 15)

                                maingame.display\_last\_discarded()

**else**:

**print**("no yellow card")

**continue**

**else**:

                    p1invalidturn = True

                    maingame.gametext\_display("Invalid move. 1 card added to hand.",2,5,15)

                    pygame.display.update()

                    mPlayer1.draw(deck, 1)

**break**  #if the user selects a card that isn't the same suit or number the card is invalidly played and they will face a draw card penalty

**else**:

**for** card **in** self.hand:

**if** card == discard:

            self.hand.remove(card)

            maingamepile.append(card) #if there are no cards in play on the main game pile then the card the user selects will be discarded as normal (as this means the user is starting the game first)

**def** showhand(self):

**print**("{} Hand is: ".format(self.name))

      h=30

**for** card **in** self.hand:

**print**(card)

          maingame.displayimage(card.image,div\_iwidth,div\_iheight-h) #display's each card in a player's hand on to the game screen

          pygame.display.update()

          h=h+25 #moves each following card down by 25 to ensure the user can see all the cards

**if** len(self.hand) == 0:

        Player1wins = True

        add\_screen()

        maingame.gametext\_display("Player1 won. Game Over", 2, 2, 40)

        maingame.gametext\_display("Score: +100 ", 2,4,40)

        maingame.updatescore(Score, unodatabase.username)

        pygame.mixer.music.load("winnermusic.mp3")

        pygame.mixer.music.play(-1)

        pygame.display.update()

        time.sleep(5)

        pygame.quit()

        os.\_exit(1)

        #once a player has no cards in their hand, Player has won, displays win screen and plays winner music

**elif** len(self.hand) == 1:

**global** t

**global** unoconfirmed

        unoconfirmed = False

        t = 10

        unoroot = Tk()

        Button(unoroot, text="Click to call uno", command=maingame.unocalled).pack()

        unoroot.mainloop()

**while** t **and** unoconfirmed == False:

            mins, secs = divmod(t, 60)

            timer = '{:02d}:{:02d}'.format(mins, secs)

**print**(timer, end="\r")

            time.sleep(1)

            t -= 1

**if** t == 0 **and** unoconfirmed == False:

            ms.showerror("","Time out! Penalty: draw a card")

            mPlayer1.draw(deck,1)

            mPlayer1.showhand()

            mPlayer2.showhand()

**elif** t == 0 **and** unoconfirmed == True:

            ms.showinfo("","UNO Called")

            time.sleep(1)

        #timer aspect not working but uno function is there

**class** Player2(Player):

**def** discard(self):

**if** len(self.hand) != 0:

            discard\_card = self.hand[maingame.down]

            self.throwAway(discard\_card) #if the player has cards left in their hand, it will discard the card based on their keyboard input using throwAway

            multiplayergame.p2turn = False

**return** multiplayergame.p2turn

**def** throwAway(self, discard):

**global** maingamepile

**if** len(maingamepile) > 1: #checks if there is a card or cards in play already

        lastcardplaced = maingamepile[-1] #if there is then the last card on the pile = the last card placed

**for** card **in** self.hand:

**if** card == discard:

**if** card.number == lastcardplaced.number **or** card.suit == lastcardplaced.suit: #checks if card selected to be discarded has the same suit or number as the last card in play on the pile

                    self.hand.remove(card) #card is removed from player's hand

                    maingamepile.append(card) #card is added on to main game pile

**break**

**elif** card.number == "+2":

                        maingame.gametext\_display("Player1 Draws 2 more cards", 2, 4,15)

                        pygame.display.update()

                        mPlayer1.draw(deck, 2) #if card is a +2, computer gets 2 more cards

                        self.hand.remove(card) #card is removed from player's hand

                        maingamepile.append(card) #card is added on to main game pile

**elif** card.number == "skip":

                        p2skipturn = True

                        maingame.gametext\_display("Player1's Turn Will Be Skipped Next Round", 2, 4,15)

                        pygame.display.update() #if card is a skip, computer's turn will be skipped next in the main game loop

                        self.hand.remove(card) #card is removed from player's hand

                        maingamepile.append(card) #card is added on to main game pile

**elif** card.number == "reverse":

                        p2reverseturn = True

                        maingame.gametext\_display("Player1's Turn Will Be Reversed Next Round", 2,4,15)

                        pygame.display.update() #if card is a reverse, computer's turn will be reversed next in the main game loop (effectively player gets another free turn)

                        self.hand.remove(card) #card is removed from player's hand

                        maingamepile.append(card) #card is added on to main game pile

**elif** card.suit == "wild":

**print**("wild card played")

                    Score = Score + 70

                    maingame.gametext\_display("Player1 draws 4 cards", 2, 4, 15)

                    #pygame.display.update()

                    mPlayer1.draw(deck, 4)

                    maingame.colourchangescreen()

**if** Red == True:

**for** card **in** self.hand:

**if** card.suit == "red":

                                self.hand.remove(card)

                                maingamepile.append(card)

                                maingame.gametext\_display("Player2 played this card", 2,5, 15)

                                maingame.display\_last\_discarded()

**else**:

**print**("no red card")

**continue**

**elif** Green == True:

**for** card **in** self.hand:

**if** card.suit == "green":

                                self.hand.remove(card)

                                maingamepile.append(card)

                                maingame.gametext\_display("Player2 played this card", 2,5, 15)

                                maingame.display\_last\_discarded()

**else**:

**print**("no green card")

**continue**

**elif** Blue == True:

**for** card **in** self.hand:

**if** card.suit == "blue":

                                self.hand.remove(card)

                                maingamepile.append(card)

                                maingame.gametext\_display("Player2 played this card", 2,5, 15)

                                maingame.display\_last\_discarded()

**else**:

**print**("no blue card")

**continue**

**elif** Yellow == True:

**for** card **in** self.hand:

**if** card.suit == "yellow":

                                self.hand.remove(card)

                                maingamepile.append(card)

                                maingame.gametext\_display("Player2 played this card", 2,5, 15)

                                maingame.display\_last\_discarded()

**else**:

**print**("no yellow card")

**continue**

**else**:

                    invalidturn = True

                    maingame.gametext\_display("Invalid move. 1 card added to hand.",2,5,15)

                    pygame.display.update()

                    mPlayer2.draw(deck, 1)

**break**  #if the user selects a card that isn't the same suit or number the card is invalidly played and they will face a draw card penalty

**else**:

**for** card **in** self.hand:

**if** card == discard:

            self.hand.remove(card)

            maingamepile.append(card) #if there are no cards in play on the main game pile then the card the user selects will be discarded as normal (as this means the user is starting the game first)

**def** showhand(self):

        #deckImg = pygame.image.load('deck\_image.png')

**print**("{}'s Hand is: ".format(self.name))

        h=30

**for** card **in** self.hand:

**print**(card)

            maingame.displayimage(card.image,div\_iwidth-600,div\_iheight-h) #parameters altered slightly to images displayed on the right hand side of the screen

            pygame.display.update()

            h=h+25

**if** len(self.hand) == 0:

            Player2wins = True

            maingame.add\_screen()

            maingame.gametext\_display("Player2 won. Game Over", 2, 2, 40)

            maingame.gametext\_display("Score: +100 ", 2,4,40)

            maingame.updatescore(Score, unodatabase.username)

            pygame.display.update()

            time.sleep(5)

            pygame.quit()

            os.\_exit(1)

            #similar to Player showhand

**elif** len(self.hand) == 1:

**global** t

**global** unoconfirmed

            unoconfirmed = False

            t = 10

            unoroot = Tk()

            Button(unoroot, text="Click to call uno", command=maingame.unocalled).pack()

            unoroot.mainloop()

**while** t > 0 **and** unoconfirmed == False:

                mins, secs = divmod(t, 60)

                timer = '{:02d}:{:02d}'.format(mins, secs)

**print**(timer, end="\r")

                time.sleep(1)

                t -= 1

**if** t == 0 **and** unoconfirmed == False:

                ms.showerror("","Time out! Penalty: draw a card")

                mPlayer2.draw(deck,1)

                mPlayer2.showhand()

                mPlayer1.showhand()

**elif** t == 0 **and** unoconfirmed == True:

                ms.showinfo("","UNO Called")

                time.sleep(1)

        #timer aspect not working but uno function is there

**class** AI(Player):

**def** discard(self):

**global** maingamepile

        lastcardplaced = maingamepile[-1]

**print**("lastplayed card is", lastcardplaced)

**for** i **in** range (len(self.hand)):

            ai\_card = self.hand[i]

**if** ai\_card.suit == lastcardplaced.suit **or** ai\_card.number == lastcardplaced.number **and** (lastcardplaced.number != "+2" **or** lastcardplaced.number != "reverse" **or** lastcardplaced.number != "skip"): #checks if card the computer wants to discard is the same suit or number as the card the player first discarded

**print**("computer discarded", ai\_card)

                self.aithrowAway(ai\_card)

**elif** ai\_card.number == "+2":

                    maingame.gametext\_display("Player1 Draws 2 more cards", 2, 5,15)

                    time.sleep(1.5)

                    pygame.display.update()

                    Player1.draw(deck, 2)

                    self.aithrowAway(ai\_card)

**elif** ai\_card.number == "skip":

                    maingame.gametext\_display("Player1's Turn Skipped", 2, 5,15)

                    time.sleep(1.5)

                    Computerskip = True

                    pygame.display.update()

                    Computer.discard()

                    self.aithrowAway(ai\_card)

**elif** ai\_card.number == "reverse":

                    maingame.gametext\_display("Player1's Turn Reversed", 2,5,15)

                    time.sleep(1.5)

                    Computerreverse = True

                    pygame.display.update()

                    Computer.discard()

                    self.aithrowAway(ai\_card)

**elif** ai\_card.suit == "wild":

                maingame.gametext\_display("Player1 Draws 4 more cards",2 ,5, 15)

                time.sleep(1.5)

                pygame.display.update()

                Player1.draw(deck, 4)

                self.aithrowAway(ai\_card)

**else**:

                maingame.gametext\_display("Computer draws a card", 2, 7, 15)

                time.sleep(1.5)

                Computer.draw(deck, 1)

                #Computer.showhand()

**break**

**break**

            #works similarly to the Player discard function. Check card suit, number and follows UNO rules accordingly

**def** aithrowAway(self, discard):

**for** card **in** self.hand:

**if** card == discard:

              self.hand.remove(card)

              maingamepile.append(card) #removes card from hand and places in main game pile

**elif** card.suit == "wild" **and** card == discard:

**if** card.number == '4x' **or** card.number == '0':

                    s = ["yellow","blue","red","green"]

                    colourchosen = random.choice(s)

**for** card **in** self.hand:

**if** card.suit == colourchosen:

                            self.hand.remove(card)

                            maingamepile.append(card)

**def** showhand(self):

        deckImg = pygame.image.load('deck\_image.png')

**print**("{}'s Hand is: ".format(self.name))

        h=30

**for** card **in** self.hand:

**print**(card)

            maingame.displayimage(deckImg,div\_iwidth-600,div\_iheight-h) #parameters altered slightly to images displayed on the right hand side of the screen

            pygame.display.update()

            h=h+25

**if** len(self.hand) == 0:

            Computerwins = True

            maingame.add\_screen()

            maingame.gametext\_display("Computer won. Game Over", 2, 2, 40)

            maingame.gametext\_display("Score: +100 ", 2,4,40)

            pygame.display.update()

            time.sleep(5)

            pygame.quit()

            os.\_exit(1)

            #similar to Player showhand

deck = Deck() #initialises deck

maingamepile = [] #initialises maingamepile

Player1 = Player('Player1')

Computer = AI('Computer')

#singleplayer stuff

#GUI + EXTRA FUNCTION STUFF

window= Tk()

**def** callback():

**global** dealnumber

**if** "." **not** **in**(einput.get()):

        dealnumber = int(einput.get())

**if** dealnumber > 15 **or** dealnumber == 0:

            ms.showerror("Error!", "Invalid Input")

**else**:

            ms.showinfo("Success", "Please close the deal setting window")

**return** dealnumber

**else**:

        ms.showerror("Error!", "Invalid Input")

cardsnumber = callback

label= Label(window, text="Please enter an amount of cards (15 or less) that you wish to start with, click save and then close this window", fg='red', font=("Helvetica", 16))

label.place(x=60, y=50)

einput = Entry(window, text="Enter start DEAL amount", bd=20)

einput.place(x=100, y=150)

button= Button(window, text="Save", fg='green', command=callback)

button.place(x=100, y=100)

window.title('Deal Setting')

window.geometry("1200x250+10+10")

window.mainloop()

window.quit()

mdeck = Deck()

mdeck.shuffle()

mPlayer1 = Playerone("Player1")

mPlayer2 = Player2("Player2")

mPlayer1.draw(mdeck , dealnumber)

mPlayer2.draw(mdeck , dealnumber)

#initialises players for multiplayer game

**class** maingame: #class for main game functionality, OOP required for multiplayer purposes

**def** \_\_init\_\_(self):

        self.iwidth = iwidth

        self.iheight = iheight

        self.width = width

        self.height = height

        self.gametext = gametext

        self.textSurface = textSurface

        self.lastcardplaced = lastcardplaced

        self.fps = fps

        self.backgroundImg = backgroundImg

        self.mouse = mouse

        self.click = click

        self.button\_font\_size = button\_font\_size

        self.down = down

        #all variables defined in my main game that aren't already globalised

        #below are the methods used in my game

**def** displayimage(image\_name,div\_iwidth, div\_iheight): #displaying imaages (UNO cards) on the screen

        maingame.iwidth = x-div\_iwidth

        maingame.iheight = y-div\_iheight

        uno\_window.blit(image\_name, (maingame.iwidth,maingame.iheight))

        time.sleep(0.3)

**def** text\_objects(text, font): #this function takes the rectangle and puts it over the whole of the text so it can be moved as one

        maingame.textSurface = font.render(text, True, black)

**return** maingame.textSurface, maingame.textSurface.get\_rect()

**def** gametext\_display(text,divby\_x,divby\_y,fontsize): #displaying text on the screen

        maingame.gametext = pygame.font.Font('freesansbold.ttf', fontsize)

        TextSurf,TextRect = maingame.text\_objects(text, maingame.gametext)

        TextRect.center = ((x/divby\_x) ,(y/divby\_y))

        uno\_window.blit(TextSurf, TextRect)

        pygame.display.update()

**def** deal\_deck(dealnumber):

        deck.shuffle()

        deck.showdeck()

        Player1.draw(deck, dealnumber)

        Player1.showhand()

        Computer.draw(deck, dealnumber)

        Computer.showhand()

        maingame.gametext\_display('Player1 starts first, use the number keys to select a card',2,12,15)

**def** unocalled():

**global** unoconfirmed

**global** t

        duration = 2000

        top = Toplevel()

        top.title('UNO alert')

        Message(top, text="UNO called!", padx=20, pady=20).pack()

        top.after(duration, top.destroy)

        unoconfirmed = True

        t=0

**def** display\_last\_discarded():

**global** maingamepile

        maingame.lastcardplaced = maingamepile[-1]

        maingame.displayimage(maingame.lastcardplaced.image, div\_iwidth-300, div\_iheight-150)

        maingame.gametext\_display("Last placed card is:", 2, 3.5, 15)

        pygame.display.update()

        #an image of the last card in play will be displayed to the screen

**def** discard\_card\_selected(): #singleplayer turn controller

**global** skipturn

**global** reverseturn

**global** computerreverse

**global** computerskip

**global** invalidturn

**if** Player1wins == False **and** Computerwins == False **and** skipturn == False **and** reverseturn == False:

            Player1.discard()

            Computer.discard()

            maingame.empty\_singleplayer\_screen()

            Player1.showhand()

            Computer.showhand()

            maingame.display\_last\_discarded()

            time.sleep(2.5)

            #If no one has won and no action cards have been played (excluded +2), gameplay will run as normal.

**elif** skipturn == True:

            Player1.discard()

            skipturn = False

            #if a player plays a skip card then skipturn will become True, when the user selects a card to play, they will be able to discard another one and the computer won't

**elif** reverseturn == True:

            Player1.discard()

            reverseturn = False #works the same as the skip section above

**elif** computerskip == True:

            Computer.discard()

            computerskip = False

**elif** computerreverse == True:

            Computer.discard()

            computerreverse = False

**elif** invalidturn == True:

            Player1.discard()

            Computer.discard()

            #once the user is punished for making an invalid move, both will be able to play cards as normal

**def** deck\_image(width,height):

**global** deckImg

        deckImg = pygame.image.load('deck\_image.png')

        uno\_window.blit(deckImg, (width,height))

        #blank uno card image to visually represent the 'pile'

**def** add\_screen():

**global** uno\_window

        uno\_window = pygame.display.set\_mode((x, y)) #creates a window with specified resolution (x,y)

        uno\_window.fill(white)

        pygame.display.set\_caption('Python UNO') #sets window title

        maingame.backgroundImg = pygame.image.load('background\_image.png')

        maingame.displayimage(maingame.backgroundImg, x, y)

        pygame.display.update()

        #used to completely update a screen (as images can't be removed from a screen in pygame)

        #acts as a blank template

**def** createbutton(button\_name,x1,y2,w1,h2,inactive\_colour,active\_colour,action=None):

        maingame.mouse = pygame.mouse.get\_pos()

        maingame.click = pygame.mouse.get\_pressed()

**if** x1+w1 > maingame.mouse[0] > x1 **and** y2+h2 > maingame.mouse[1] > y2:

            pygame.draw.rect(uno\_window, active\_colour,(x1,y2,w1,h2))

**if** maingame.click[0] == 1 **and** action != None:

                action()

**else**:

            pygame.draw.rect(uno\_window, inactive\_colour,(x1,y2,w1,h2))

        #checks if mouse position is within button's defined region and if so, colour will change, if the button is clicked then the defined action will be performed

        maingame.button\_font\_size = pygame.font.Font("freesansbold.ttf",20)

        textSurf, textRect = maingame.text\_objects(button\_name, maingame.button\_font\_size)

        textRect.center = ( (x1+(w1/2)), (y2+(h2/2)) )

        uno\_window.blit(textSurf, textRect)

        #defines button text font and size and adds it the surface of the screen

**def** startup\_menu():

        maingame.add\_screen()

        pygame.mixer.music.load("Menumusic.mp3")

        pygame.mixer.music.play(-1)

        startup = True

**while** startup:

**for** event **in** pygame.event.get():

**if** event.type == pygame.QUIT:

                    pygame.quit()

                    quit()

                    #if the user clicks the x on the window the window will close and the game will quit

            maingame.gametext\_display('Welcome to UNO',2,2,50)

            singeplayer\_button = maingame.createbutton('SINGLEPLAYER',50,450,160,50,green,orange,maingame.singleplayer)

            mutliplayer\_button = maingame.createbutton('MULTILPLAYER',250,450,160,50,blue,orange,multiplayergame.multiplayer\_startup\_screen)

            quit\_button = maingame.createbutton('QUIT',450,450,100,50,red,orange,maingame.quitgame)

            help\_button = maingame.createbutton('HELP', 600,450,100, 50, white, orange,maingame.help\_screen)

            pygame.display.update()

        #main menu - plays main menu music and displays singleplayer,multiplaer,quit and help button

**def** quitgame():

        pygame.quit()

        quit()

        #action for quit button to exit game

**def** help\_screen():

        maingame.add\_screen()

        startup = True

**while** startup:

**for** event **in** pygame.event.get():

**if** event.type == pygame.QUIT:

                    pygame.quit()

                    quit()

**elif** event.type == pygame.KEYDOWN **and** event.key == pygame.K\_BACKSPACE:

                    maingame.startup\_menu()

            maingame.rulescreenImg = pygame.image.load('rulescreen.png')

            maingame.displayimage(maingame.rulescreenImg, x, y)

            #back\_button = maingame.createbutton('BACK',350,550,100,40,white, orange,maingame.startup\_menu)

            pygame.display.update()

            #action for help button, displays back button to main menu and rule button to rule screen and keybinds

**def** singleplayer():

        start\_time = 300

        frame\_count = 0

        frame\_rate = 5

        clock = pygame.time.Clock()

        pygame.mixer.music.load("gamemusic1.mp3")

        pygame.mixer.music.play(-1)

        maingame.empty\_singleplayer\_screen()

        #maingame.gametext\_display('Total Game time: 5 minute', 2,8,30)

        time.sleep(1)

        maingame.deal\_deck(dealnumber)

        play = True

**while** play == True:

**for** event **in** pygame.event.get():

**if** event.type == pygame.QUIT:

                    pygame.quit()

                    quit()

**elif** event.type == pygame.KEYDOWN **and** event.key == pygame.K\_BACKSPACE:

                    maingame.startup\_menu()

**elif** event.type == pygame.KEYDOWN **and** event.key == pygame.K\_q:

**print**("player forfeited")

                    Computerwins = True

                    maingame.add\_screen()

                    maingame.gametext\_display("Computer won. Game Over", 2, 2, 40)

                    maingame.gametext\_display("Score: +100 ", 2,4,40)

                    pygame.display.update()

                    time.sleep(5)

                    pygame.quit()

                    os.\_exit(1)

**elif** event.type == pygame.MOUSEBUTTONDOWN:

                    mouse\_pos = pygame.mouse.get\_pos()

**print**(mouse\_pos)

**print**("clicking")

**if** mouse\_pos[0] > 49 **and** mouse\_pos[0] < 130:

**if** mouse\_pos[1] >= 80 **and** mouse\_pos[1] < 105:

                            maingame.down = 0

**print**(maingame.down)

**print**("first card played")

                            maingame.discard\_card\_selected()

**elif** mouse\_pos[1] >= 108 **and** mouse\_pos[1] < 128:

                            maingame.down = 1

**print**(maingame.down)

                            maingame.discard\_card\_selected()

**print**("second card played")

**elif** mouse\_pos[1] >= 134 **and** mouse\_pos[1] < 153:

                            maingame.down = 2

**print**(maingame.down)

                            maingame.discard\_card\_selected()

**elif** mouse\_pos[1] >= 159 **and** mouse\_pos[1] < 177:

                            maingame.down = 3

**print**(maingame.down)

                            maingame.discard\_card\_selected()

**elif** mouse\_pos[1] >= 182 **and** mouse\_pos[1] < 203:

                            maingame.down = 4

**print**(maingame.down)

                            maingame.discard\_card\_selected()

**elif** mouse\_pos[1] >= 208 **and** mouse\_pos[1] < 228:

                            maingame.down = 5

**print**(maingame.down)

                            maingame.discard\_card\_selected()

**elif** mouse\_pos[1] >= 232 **and** mouse\_pos[1] < 251:

                            maingame.down = 6

**print**(maingame.down)

                            maingame.discard\_card\_selected()

**elif** mouse\_pos[1] >= 259 **and** mouse\_pos[1] < 276:

                            maingame.down = 7

**print**(maingame.down)

                            maingame.discard\_card\_selected()

**elif** mouse\_pos[1] >= 285 **and** mouse\_pos[1] < 303:

                            maingame.down = 8

**print**(maingame.down)

                            maingame.discard\_card\_selected()

**elif** mouse\_pos[1] >= 307 **and** mouse\_pos[1] < 329:

                            maingame.down = 9

**print**(maingame.down)

                            maingame.discard\_card\_selected()

**elif** mouse\_pos[1] >= 332 **and** mouse\_pos[1] < 352:

                            maingame.down = 10

**print**(maingame.down)

                            maingame.discard\_card\_selected()

**elif** mouse\_pos[1] >= 358 **and** mouse\_pos[1] < 378:

                            maingame.down = 11

**print**(maingame.down)

                            maingame.discard\_card\_selected()

**elif** mouse\_pos[1] >= 383 **and** mouse\_pos[1] < 403:

                            maingame.down = 12

**print**(maingame.down)

                            maingame.discard\_card\_selected()

**elif** mouse\_pos[1] >= 408 **and** mouse\_pos[1] < 427:

                            maingame.down = 13

**print**(maingame.down)

                            maingame.discard\_card\_selected()

**elif** mouse\_pos[1] >= 434 **and** mouse\_pos[1] < 546:

                            maingame.down = 14

**print**(maingame.down)

                            maingame.discard\_card\_selected()

**elif** mouse\_pos[0] > 347 **and** mouse\_pos[0] < 433 **and** mouse\_pos[1] > 201 **and** mouse\_pos[1] < 324:

**print**("central deck clicked")

**if** len(Player1.hand) <= 15:

                            Player1.draw(deck, 1)

                            Player1.showhand()

                            maingame.gametext\_display("You've drawn a card from the pile",2,5,15)

                            Computer.discard()

                            Computer.showhand()

**elif** len(Player1.hand) > 15:

                            ms.showerror("UNO Alert", "Max cards drawn please play a card or press Q to forfeit")

**elif** event.type == pygame.KEYDOWN **and** event.key == pygame.K\_1:

                    maingame.down = 0

                    maingame.discard\_card\_selected()

**elif** event.type == pygame.KEYDOWN **and** event.key == pygame.K\_2:

                    maingame.down = 1

                    maingame.discard\_card\_selected()

**elif** event.type == pygame.KEYDOWN **and** event.key == pygame.K\_3:

                    maingame.down = 2

                    maingame.discard\_card\_selected()

**elif** event.type == pygame.KEYDOWN **and** event.key == pygame.K\_4:

                    maingame.down = 3

                    maingame.discard\_card\_selected()

**elif** event.type == pygame.KEYDOWN **and** event.key == pygame.K\_5:

                    maingame.down = 4

                    maingame.discard\_card\_selected()

**elif** event.type == pygame.KEYDOWN **and** event.key == pygame.K\_6:

                    maingame.down = 5

                    maingame.discard\_card\_selected()

**elif** event.type == pygame.KEYDOWN **and** event.key == pygame.K\_7:

                    maingame.down = 6

                    maingame.discard\_card\_selected()

**elif** event.type == pygame.KEYDOWN **and** event.key == pygame.K\_8:

                    maingame.down = 7

                    maingame.discard\_card\_selected()

**elif** event.type == pygame.KEYDOWN **and** event.key == pygame.K\_9:

                    maingame.down = 8

                    maingame.discard\_card\_selected()

**elif** event.type == pygame.KEYDOWN **and** event.key == pygame.K\_0:

                    maingame.down = 9

                    maingame.discard\_card\_selected()

**elif** event.type == pygame.KEYDOWN **and** event.key == pygame.K\_s:

                    maingame.display\_last\_discarded()

                    time.sleep(4)

**elif** event.type == pygame.KEYDOWN **and** event.key == pygame.K\_SPACE:

**if** len(Player1.hand) <= 15:

                        Player1.draw(deck, 1)

                        Player1.showhand()

                        maingame.gametext\_display("You've drawn a card from the pile",2,5,15)

                        Computer.discard()

                        Computer.showhand()

**elif** len(Player1.hand) > 15:

                        ms.showerror("UNO Alert", "Max cards drawn please play a card or press Q to forfeit")

                #checks what number in the list of cards the user has selected to remove, and removes that card from their hand

                #pressing space draws a card for the user at any point in the game but only up to a maximum of 15 cards

                #pressing s displays the last card in play for 4 seconds as a reminder

            total\_seconds = start\_time - (frame\_count // frame\_rate)

**if** total\_seconds < 0:

**print**("time up")

                total\_seconds = 0

                play = False

                maingame.add\_screen()

                maingame.gametext\_display("Times up! Game over! Game tied!", 2,2,30)

                pygame.display.update()

                time.sleep(4)

                pygame.quit()

                quit()

            frame\_count += 1

            clock.tick(frame\_rate)

            #pygame.display.update()

            maingame.gametext\_display('Player1', 12, 12, 15)

            maingame.gametext\_display('Computer', 1.2, 12, 15)

            maingame.width = (x/2.3) #location on screen

            maingame.height = (y/3) #location on screen

            maingame.deck\_image(maingame.width,maingame.height) #blank UNO CARD image to represent pile

            pygame.display.update()

**def** updatescore(self, score, username):

        with sqlite3.connect('uno\_user\_database.db') as db:

            c = db.cursor()

            update\_highscore = 'UPDATE users SET userscore = ? WHERE username = ? AND userscore < ?'

            c.execute(update\_highscore,[(score),(username),(score)])

**def** colourchangescreen():

**global** colouroption

        colouroption = False

**while** colouroption == False:

**for** event **in** pygame.event.get():

**if** event.type == pygame.KEYDOWN **and** event.key == pygame.K\_r:

                    maingame.redselected()

**elif** event.type == pygame.KEYDOWN **and** event.key == pygame.K\_b:

                    maingame.blueselected()

**elif** event.type == pygame.KEYDOWN **and** event.key == pygame.K\_g:

                    maingame.greenselected()

**elif** event.type == pygame.KEYDOWN **and** event.key == pygame.K\_y:

                    maingame.greenselected()

            maingame.add\_screen()

            redbutton = maingame.createbutton('RED',50,450,160,50,red,orange,maingame.redselected)

            bluebutton = maingame.createbutton('BLUE',250,450,160,50,blue,orange,maingame.blueselected)

            greenbutton = maingame.createbutton('GREEN',450,450,100,50,green,orange,maingame.greenselected)

            yellowbutton = maingame.createbutton('YELLOW', 600,450,100, 50, yellow, orange,maingame.yellowselected)

            pygame.display.update()

**def** redselected():

**global** colouroption

**global** Red

        maingame.empty\_singleplayer\_screen()

        Red = True

        colouroption = True

**def** blueselected():

**global** colouroption

**global** Blue

        maingame.empty\_singleplayer\_screen()

        Blue = True

        colouroption = True

**def** greenselected():

**global** colouroption

**global** Green

        maingame.empty\_singleplayer\_screen()

        Green = True

        colouroption = True

**def** yellowselected():

**global** colouroption

**global** Yellow

        maingame.empty\_singleplayer\_screen()

        Yellow = True

        colouroption = True

**def** empty\_singleplayer\_screen():

**global** uno\_window

        uno\_window = pygame.display.set\_mode((x, y)) #creates a window with specified resolution (x,y)

        uno\_window.fill(white)

        pygame.display.set\_caption('Python UNO') #sets window title

        maingame.fps = pygame.time.Clock() #creates a clock that counts fps

        maingame.fps.tick(5)

        maingame.gamescreenImg = pygame.image.load('gamescreen.png')

        maingame.displayimage(maingame.gamescreenImg, x, y)

        pygame.display.update()

        maingame.gametext\_display('Player1', 12, 12, 15)

        maingame.gametext\_display('Computer', 1.2, 12, 15)

        maingame.width = (x/2.3) #location on screen

        maingame.height = (y/3) #location on screen

        maingame.deck\_image(maingame.width,maingame.height)

        #blank template for singleplayer screen, used after cards have been discarded to refresh view

**def** uno\_gui():

        maingame.startup\_menu()

**class** Multiplayeronline(maingame): #still currently working on

**def** \_\_init\_\_(self):

        self.firstplayerobject = None

        self.secondplayerobject = None

        self.multiplayeronline=False

        self.down = down

**def** empty\_multiplayer\_online\_screen():

        maingame.add\_screen()

        maingame.gametext\_display('Player1', 12, 12, 15)

        maingame.gametext\_display('Player2', 1.2, 12, 15)

        maingame.width = (x/2.3)

        maingame.height = (y/3)

        maingame.deck\_image(maingame.width,maingame.height)

**def** MOgameplay():

        Multiplayeronline.main()

**print**("Multiplayer game started")

        maingame.add\_screen()

        #global multiplayeronline

        multiplayeronline = True

        Multiplayeronline.firstplayerobject.draw(deck, 5)

        Multiplayeronline.secondplayerobject.draw(deck, 5)

        Multiplayeronline.firstplayerobject.showhand()

**while** multiplayeronline ==  True:

**for** event **in** pygame.event.get():

**if** event.type == pygame.QUIT:

                    pygame.quit()

                    quit()

**elif** event.type == pygame.KEYDOWN **and** event.key == pygame.K\_1:

                    Multiplayeronline.down = 0

                    Multiplayeronline.gamemanager()

**elif** event.type == pygame.KEYDOWN **and** event.key == pygame.K\_2:

                    Multiplayeronline.down = 1

                    multiplayergame.gamemanager()

**elif** event.type == pygame.KEYDOWN **and** event.key == pygame.K\_3:

                    Multiplayeronline.down = 2

                    Multiplayeronline.main()

**elif** event.type == pygame.KEYDOWN **and** event.key == pygame.K\_4:

                    Multiplayeronline.down = 3

                    Multiplayeronline.main()

**elif** event.type == pygame.KEYDOWN **and** event.key == pygame.K\_5:

                    Multiplayeronline.down = 4

                    Multiplayeronline.main()

**elif** event.type == pygame.KEYDOWN **and** event.key == pygame.K\_6:

                    Multiplayeronline.down = 5

                    Multiplayeronline.main()

**elif** event.type == pygame.KEYDOWN **and** event.key == pygame.K\_7:

                    Multiplayeronline.down = 6

                    Multiplayeronline.main()

**elif** event.type == pygame.KEYDOWN **and** event.key == pygame.K\_8:

                    Multiplayeronline.down = 7

                    Multiplayeronline.main()

**elif** event.type == pygame.KEYDOWN **and** event.key == pygame.K\_9:

                    Multiplayeronline.down = 8

                    Multiplayeronline.main()

**elif** event.type == pygame.KEYDOWN **and** event.key == pygame.K\_0:

                    Multiplayeronline.down = 9

                    Multiplayeronline.main()

**elif** event.type == pygame.KEYDOWN **and** event.key == pygame.K\_s:

                    Multiplayeronline.display\_last\_discarded()

                    time.sleep(4)

            maingame.gametext\_display('Player1', 12, 12, 15)

            maingame.gametext\_display('Player2', 1.2, 12, 15)

            maingame.width = (x/2.3)

            maingame.height = (y/3)

            maingame.deck\_image(maingame.width,maingame.height)

            pygame.display.update()

**def** gamemanager(self):

        Multiplayeronline.firstplayerobject.showhand()

        Multiplayeronline.secondplayerobject.showhand()

**def** MOwinnerscreen(playerthatwon):

        maingame.add\_screen()

        maingame.gametext\_display(playerthatwon," won. Game Over", 2, 2, 40)

        gametext\_display("Score: +100 ", 2,4,40)

        pygame.mixer.music.load("winnermusic.mp3")

        pygame.mixer.music.play(-1)

        pygame.display.update()

        time.sleep(5)

        pygame.quit()

        os.\_exit(1)

        #once player has no cards in their hand, Player has won, displays win screen and plays winner music

**def** main(): #ADD USER GAMEPLAY BETWEEN SERVERS AND THEN TEST EVERYTHING

        run = True

        pygame.mixer.music.load("gamemusic1.mp3")

        pygame.mixer.music.play(-1)

        host = socket.gethostname()

        ip = socket.gethostbyname(host)

        n = Network(ip)

        p = n.getID()

**print**(p)

        Multiplayeronline.firstplayerobject = p

**while** run:

            player2 = n.send(p)

            Multiplayeronline.secondplayerobject = player2

**for** event **in** pygame.event.get():

**if** event.type == pygame.QUIT:

                    run = False

                    pygame.quit()

**if** Multiplayeronline.firstplayerobject.won == True:

                run = False

                Multiplayeronline.MOwinnerscreen(Multiplayeronline.firstplayerobject)

**elif** Multiplayeronline.secondplayerobject.won == True:

                run = False

                Multiplayeronline.MOwinnerscreen(Multiplayeronline.secondplayerobject)

**else**:

                Multiplayeronline.MOgameplay()

            pygame.display.update()

**class** multiplayergame(maingame): #class for multiplayer including main game functionality

**def** \_\_init\_\_(self):

        self.deck = Deck()

        self.maingamepile = []

        self.p1turn = False

        self.p2turn = False

**def** multiplayer\_startup\_screen():

        maingame.add\_screen()

        multiplayer\_selected = True

**while** multiplayer\_selected == True:

**for** event **in** pygame.event.get():

**if** event.type == pygame.QUIT:

                    pygame.quit()

                    quit()

            multiplayer\_local\_button = maingame.createbutton('Multiplayer Local', 150,450,160,50,blue,orange,multiplayergame.multiplayer\_gui)

            multiplayer\_online\_button = maingame.createbutton('Multiplayer Online', 450,450,160,50,red,orange,Multiplayeronline.MOgameplay)

            pygame.display.update()

**def** empty\_multiplayer\_screen():

**global** uno\_window

        uno\_window = pygame.display.set\_mode((x, y)) #creates a window with specified resolution (x,y)

        uno\_window.fill(white)

        pygame.display.set\_caption('Python UNO') #sets window title

        maingame.fps = pygame.time.Clock() #creates a clock that counts fps

        maingame.fps.tick(5)

        maingame.gamescreenImg = pygame.image.load('gamescreen.png')

        maingame.displayimage(maingame.gamescreenImg, x, y)

        maingame.gametext\_display('Player1', 12, 12, 15)

        maingame.gametext\_display('Player2', 1.2, 12, 15)

        maingame.width = (x/2.3)

        maingame.height = (y/3)

        maingame.deck\_image(maingame.width,maingame.height)

        mPlayer1.showhand()

        mPlayer2.showhand()

**def** multiplayer\_gui():

**global** mdeck

**global** mPlayer1

**global** mPlayer2

        pygame.mixer.music.load("gamemusic1.mp3")

        pygame.mixer.music.play(-1)

        multiplayergame.empty\_multiplayer\_screen()

        maingame.gametext\_display('Player1 starts first', 2, 12,15)

        multiplayergame.p1turn = True

        multiplayergame.p2turn = False

**global** start

        start = True

**while** start == True:

**for** event **in** pygame.event.get():

**if** event.type == pygame.QUIT:

                    pygame.quit()

                    quit()

**elif** event.type == pygame.KEYDOWN **and** event.key == pygame.K\_BACKSPACE:

                    maingame.startup\_menu()

**elif** event.type == pygame.MOUSEBUTTONDOWN:

                    mouse\_pos = pygame.mouse.get\_pos()

**print**(mouse\_pos)

**print**("clicking")

                    #print(cardpos)

**if** mouse\_pos[0] > 49 **and** mouse\_pos[0] < 130:

**if** mouse\_pos[1] >= 80 **and** mouse\_pos[1] < 105:

                            maingame.down = 0

**print**(maingame.down)

                            multiplayergame.gamemanager()

**elif** mouse\_pos[1] >= 108 **and** mouse\_pos[1] < 128:

                            maingame.down = 1

**print**(maingame.down)

                            multiplayergame.gamemanager()

**elif** mouse\_pos[1] >= 134 **and** mouse\_pos[1] < 153:

                            maingame.down = 2

**print**(maingame.down)

                            multiplayergame.gamemanager()

**elif** mouse\_pos[1] >= 159 **and** mouse\_pos[1] < 177:

                            maingame.down = 3

**print**(maingame.down)

                            multiplayergame.gamemanager()

**elif** mouse\_pos[1] >= 182 **and** mouse\_pos[1] < 203:

                            maingame.down = 4

**print**(maingame.down)

                            multiplayergame.gamemanager()

**elif** mouse\_pos[1] >= 208 **and** mouse\_pos[1] < 228:

                            maingame.down = 5

**print**(maingame.down)

                            multiplayergame.gamemanager()

**elif** mouse\_pos[1] >= 232 **and** mouse\_pos[1] < 251:

                            maingame.down = 6

**print**(maingame.down)

                            multiplayergame.gamemanager()

**elif** mouse\_pos[1] >= 259 **and** mouse\_pos[1] < 276:

                            maingame.down = 7

**print**(maingame.down)

                            multiplayergame.gamemanager()

**elif** mouse\_pos[1] >= 285 **and** mouse\_pos[1] < 303:

                            maingame.down = 8

**print**(maingame.down)

                            multiplayergame.gamemanager()

**elif** mouse\_pos[1] >= 307 **and** mouse\_pos[1] < 329:

                            maingame.down = 9

**print**(maingame.down)

                            multiplayergame.gamemanager()

**elif** mouse\_pos[1] >= 332 **and** mouse\_pos[1] < 352:

                            maingame.down = 10

**print**(maingame.down)

                            multiplayergame.gamemanager()

**elif** mouse\_pos[1] >= 358 **and** mouse\_pos[1] < 378:

                            maingame.down = 11

**print**(maingame.down)

                            multiplayergame.gamemanager()

**elif** mouse\_pos[1] >= 383 **and** mouse\_pos[1] < 403:

                            maingame.down = 12

**print**(maingame.down)

                            multiplayergame.gamemanager()

**elif** mouse\_pos[1] >= 408 **and** mouse\_pos[1] < 427:

                            maingame.down = 13

**print**(maingame.down)

                            multiplayergame.gamemanager()

**elif** mouse\_pos[1] >= 434 **and** mouse\_pos[1] < 546:

                            maingame.down = 14

**print**(maingame.down)

                            multiplayergame.gamemanager()

**elif** mouse\_pos[0] > 650 **and** mouse\_pos[0] < 728:

**if** mouse\_pos[1] >= 80 **and** mouse\_pos[1] < 105:

                            maingame.down = 0

**print**(maingame.down)

                            multiplayergame.gamemanager()

**elif** mouse\_pos[1] >= 108 **and** mouse\_pos[1] < 128:

                            maingame.down = 1

**print**(maingame.down)

                            multiplayergame.gamemanager()

**elif** mouse\_pos[1] >= 134 **and** mouse\_pos[1] < 153:

                            maingame.down = 2

**print**(maingame.down)

                            multiplayergame.gamemanager()

**elif** mouse\_pos[1] >= 159 **and** mouse\_pos[1] < 177:

                            maingame.down = 3

**print**(maingame.down)

                            multiplayergame.gamemanager()

**elif** mouse\_pos[1] >= 182 **and** mouse\_pos[1] < 203:

                            maingame.down = 4

**print**(maingame.down)

                            multiplayergame.gamemanager()

**elif** mouse\_pos[1] >= 208 **and** mouse\_pos[1] < 228:

                            maingame.down = 5

**print**(maingame.down)

                            multiplayergame.gamemanager()

**elif** mouse\_pos[1] >= 232 **and** mouse\_pos[1] < 251:

                            maingame.down = 6

**print**(maingame.down)

                            multiplayergame.gamemanager()

**elif** mouse\_pos[1] >= 259 **and** mouse\_pos[1] < 276:

                            maingame.down = 7

**print**(maingame.down)

                            multiplayergame.gamemanager()

**elif** mouse\_pos[1] >= 285 **and** mouse\_pos[1] < 303:

                            maingame.down = 8

**print**(maingame.down)

                            multiplayergame.gamemanager()

**elif** mouse\_pos[1] >= 307 **and** mouse\_pos[1] < 329:

                            maingame.down = 9

**print**(maingame.down)

                            multiplayergame.gamemanager()

**elif** mouse\_pos[1] >= 332 **and** mouse\_pos[1] < 352:

                            maingame.down = 10

**print**(maingame.down)

                            multiplayergame.gamemanager()

**elif** mouse\_pos[1] >= 358 **and** mouse\_pos[1] < 378:

                            maingame.down = 11

**print**(maingame.down)

                            multiplayergame.gamemanager()

**elif** mouse\_pos[1] >= 383 **and** mouse\_pos[1] < 403:

                            maingame.down = 12

**print**(maingame.down)

                            multiplayergame.gamemanager()

**elif** mouse\_pos[1] >= 408 **and** mouse\_pos[1] < 427:

                            maingame.down = 13

**print**(maingame.down)

                            multiplayergame.gamemanager()

**elif** mouse\_pos[1] >= 434 **and** mouse\_pos[1] < 546:

                            maingame.down = 14

**print**(maingame.down)

                            multiplayergame.gamemanager()

**elif** event.type == pygame.KEYDOWN **and** event.key == pygame.K\_1:

                    maingame.down = 0

                    multiplayergame.gamemanager()

**elif** event.type == pygame.KEYDOWN **and** event.key == pygame.K\_2:

                    maingame.down = 1

                    multiplayergame.gamemanager()

**elif** event.type == pygame.KEYDOWN **and** event.key == pygame.K\_3:

                    maingame.down = 2

                    multiplayergame.gamemanager()

**elif** event.type == pygame.KEYDOWN **and** event.key == pygame.K\_4:

                    maingame.down = 3

                    multiplayergame.gamemanager()

**elif** event.type == pygame.KEYDOWN **and** event.key == pygame.K\_5:

                    maingame.down = 4

                    multiplayergame.gamemanager()

**elif** event.type == pygame.KEYDOWN **and** event.key == pygame.K\_6:

                    maingame.down = 5

                    multiplayergame.gamemanager()

**elif** event.type == pygame.KEYDOWN **and** event.key == pygame.K\_7:

                    maingame.down = 6

                    multiplayergame.gamemanager()

**elif** event.type == pygame.KEYDOWN **and** event.key == pygame.K\_8:

                    maingame.down = 7

                    multiplayergame.gamemanager()

**elif** event.type == pygame.KEYDOWN **and** event.key == pygame.K\_9:

                    maingame.down = 8

                    multiplayergame.gamemanager()

**elif** event.type == pygame.KEYDOWN **and** event.key == pygame.K\_0:

                    maingame.down = 9

                    multiplayergame.gamemanager()

**elif** event.type == pygame.KEYDOWN **and** event.key == pygame.K\_s:

                    multiplayergame.display\_last\_discarded()

                    time.sleep(4)

**elif** event.type == pygame.KEYDOWN **and** event.key == pygame.K\_SPACE:

**if** multiplayergame.p1turn == True:

                        mPlayer1.draw(deck, 1)

                        multiplayergame.gametext\_display("Player1 draw's a card", 2, 5, 15)

                        pygame.display.update()

                        multiplayergame.empty\_multiplayer\_screen()

**elif** multiplayergame.p2turn == True:

                        mPlayer2.draw(deck, 1)

                        multiplayergame.gametext\_display("Player2 draw's a card",2,5,15)

                        pygame.display.update()

                        multiplayergame.empty\_multiplayer\_screen()

                #checks what number in the list of cards the user has selected to remove, and removes that card from their hand

                #pressing space draws a card for the user at any point in the game

                #pressing s displays the last card in play for 4 seconds as a reminder

            maingame.gametext\_display('Player1', 12, 12, 15)

            maingame.gametext\_display('Player2', 1.2, 12, 15)

            maingame.width = (x/2.3) #location on screen

            maingame.height = (y/3) #location on screen

            maingame.deck\_image(maingame.width,maingame.height) #blank UNO CARD image to represent pile

            pygame.display.update()

        #multiplayer local main screen and functionality

**def** gamemanager():

**global** p1reverseturn

**global** p1skipturn

**global** p2reverseturn

**global** p2skipturn

        #global netconnection

**if** start == True:

**if** multiplayergame.p1turn == True **and** multiplayergame.p2turn == False **and** p1reverseturn == False **and** p1skipturn == False **and** p2reverseturn == False **and** p2skipturn == False:

                maingame.gametext\_display("Player1's turn", 2,5,15)

                pygame.display.update()

                mPlayer1.discard()

                multiplayergame.p1turn = False

                multiplayergame.p2turn = True

                maingame.gametext\_display("End of Player1's Turn", 2,3,15)

                pygame.display.update()

                multiplayergame.empty\_multiplayer\_screen()

**elif** multiplayergame.p2turn == True **and** multiplayergame.p1turn == False **and** p1reverseturn == False **and** p1skipturn == False **and** p2reverseturn == False **and** p2skipturn == False:

                maingame.gametext\_display("Player2's turn", 2,5,15)

                pygame.display.update()

                mPlayer2.discard()

                multiplayergame.p1turn = True

                maingame.gametext\_display("End of Player2's Turn", 2,3,15)

                pygame.display.update()

                multiplayergame.empty\_multiplayer\_screen()

**elif** multiplayergame.p1turn == True **and** multiplayergame.p2turn == False **and** (p1reverseturn == True **or** p1skipturn == True):

                mPlayer1.discard()

**elif** multiplayergame.p2turn == True **and** multiplayergame.p1turn == False **and** (p2reverseturn == True **or** p2skipturn == True):

                mPlayer2.discard()

        #manages who''s turn it is between each player and what certain playing cards result in

#maingame.uno\_gui()

# make database and users (if not exists already) table at programme start up

with sqlite3.connect('uno\_user\_database.db') as db:

    c = db.cursor()

c.execute('CREATE TABLE IF NOT EXISTS users (user\_id INTEGER PRIMARY KEY, username TEXT NOT NULL ,password TEXT NOT NULL, userscore INTEGER);')

db.commit()

db.close()

#creates a table with 4 columns, userid, username, password and score

#login class for uno

**class** Unodatabase:

**def** \_\_init\_\_(self,master):

        # Window

        self.master = master

        # Some Usefull variables

        self.username = StringVar()

        self.password = StringVar()

        self.userscore = IntVar()

        self.n\_username = StringVar()

        self.n\_password = StringVar()

        #Create Widgets

        self.widgets()

    #Login Function

**def** login(self):

        #Establish Connection

        with sqlite3.connect('uno\_user\_database.db') as db:

            c = db.cursor()

        #Find user if there is any, if username and password exist, log in else username not found

        find\_user = ('SELECT \* FROM users WHERE username = ? and password = ?')

        c.execute(find\_user,[(self.username.get()),(self.password.get())])

        result = c.fetchall()

**if** result:

            self.logf.pack\_forget()

            self.head['text'] = self.username.get() + '\n Logged In'

            self.head['pady'] = 100

            self.head['padx'] = 100

**global** login

            login = True

            maingame.uno\_gui()

**else**:

            ms.showerror('Error!','Username Not Found.')

**def** new\_user(self):

        #Establish Connection

        with sqlite3.connect('uno\_user\_database.db') as db:

            c = db.cursor()

        #Find Existing username if user enters name that already exists, they must try another one else success

        find\_user = ('SELECT \* FROM users WHERE username = ?')

        c.execute(find\_user,[(self.username.get())])

**if** c.fetchall():

            ms.showerror('Error!','Username Taken Try a Diffrent One.')

**else**:

            ms.showinfo('Success!','Account Created')

            self.log()

        #Create New Account

        insert = 'INSERT INTO users(username,password) VALUES(?,?)'

        c.execute(insert,[(self.n\_username.get()),(self.n\_password.get())])

        db.commit()

        #Frame Packing Methords

**def** log(self):

        self.username.set('')

        self.password.set('')

        self.crf.pack\_forget()

        self.head['text'] = 'LOGIN'

        self.logf.pack()

**def** cr(self):

        self.n\_username.set('')

        self.n\_password.set('')

        self.logf.pack\_forget()

        self.head['text'] = 'Create Account'

        self.crf.pack()

    #Draw Widgets

**def** widgets(self):

        self.head = Label(self.master,text = 'LOGIN',font = ('',35),pady = 10)

        self.head.pack()

        self.logf = Frame(self.master,padx =10,pady = 10)

        Label(self.logf,text = 'Username: ',font = ('',20),pady=5,padx=5).grid(sticky = W)

        Entry(self.logf,textvariable = self.username,bd = 5,font = ('',15)).grid(row=0,column=1)

        Label(self.logf,text = 'Password: ',font = ('',20),pady=5,padx=5).grid(sticky = W)

        Entry(self.logf,textvariable = self.password,bd = 5,font = ('',15),show = '\*').grid(row=1,column=1)

        Button(self.logf,text = ' Login ',bd = 3 ,font = ('',15),padx=5,pady=5,command=self.login).grid()

        Button(self.logf,text = ' Create Account ',bd = 3 ,font = ('',15),padx=5,pady=5,command=self.cr).grid(row=2,column=1)

        self.logf.pack()

        self.crf = Frame(self.master,padx =10,pady = 10)

        Label(self.crf,text = 'Username: ',font = ('',20),pady=5,padx=5).grid(sticky = W)

        Entry(self.crf,textvariable = self.n\_username,bd = 5,font = ('',15)).grid(row=0,column=1)

        Label(self.crf,text = 'Password: ',font = ('',20),pady=5,padx=5).grid(sticky = W)

        Entry(self.crf,textvariable = self.n\_password,bd = 5,font = ('',15),show = '\*').grid(row=1,column=1)

        Button(self.crf,text = 'Create Account',bd = 3 ,font = ('',15),padx=5,pady=5,command=self.new\_user).grid()

        Button(self.crf,text = 'Go to Login',bd = 3 ,font = ('',15),padx=5,pady=5,command=self.log).grid(row=2,column=1)

    '''''def updatescore(self,score):

        with sqlite3.connect('uno\_user\_database.db') as db:

            c = db.cursor()

            update\_score = ('UPDATE users SET userscore ='''

#create log in window and application object

root = Tk()

root.title("Login")

Unodatabase(root)

root.mainloop()