PRO-C205

GAME WINDOW

What is our GOAL for this CLASS?

In this class, we have created a game window using Tkinter and we also learned to create a separate board for each player.

What did we ACHIEVE in the class TODAY?

- Created a game window using tkinter.
- Added the left board and right board for two player.bodies.
- Added dice to the game.

Which CONCEPTS/ CODING BLOCKS did we cover today?

Add a bulleted list of new coding concepts that were covered in the class.

- GUI using tkinter.
- Socket programming.

How did we DO the activities?

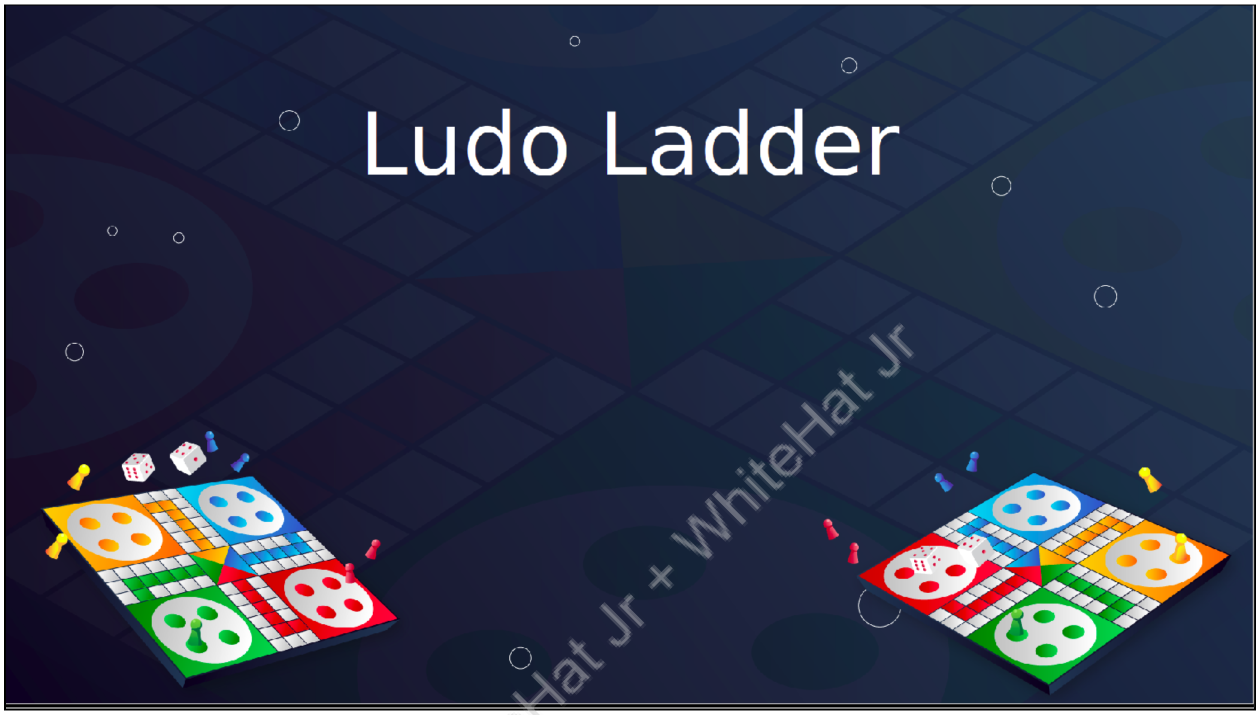
In earlier classes we have learned to create multiplayer games using a database where all the records of players were kept in the database.

Today, you created a game window which would be the playground for th.

1. Write a function called **gameWindow().** Inside this function create a GUI using Tkinter. And call this function inside the **saveName().**

```
def gameWindow():
   global gameWindow
   global canvas2
   global screen_width
   global screen_height
   global dice

   gameWindow = Tk()
   gameWindow.title("Ludo Ladder")
   gameWindow.attributes('-fullscreen',True)

   screen_width = gameWindow.winfo_screenwidth()
   screen_height = gameWindow.winfo_screenheight()

   bg = ImageTk.PhotoImage(file = "./assets/background.png")

   canvas2 = Canvas( gameWindow, width = 500,height = 500)
   canvas2.pack(fill = "both", expand = True)

   # Display image
   canvas2.create_image( 0, 0, image = bg, anchor = "nw")

   # Add Text
   canvas2.create_text( screen_width/2, screen_height/5, text =
"Ludo Ladder", font=("Chalkboard SE",100), fill="white")
```

```
    gameWindow.mainloop()
```

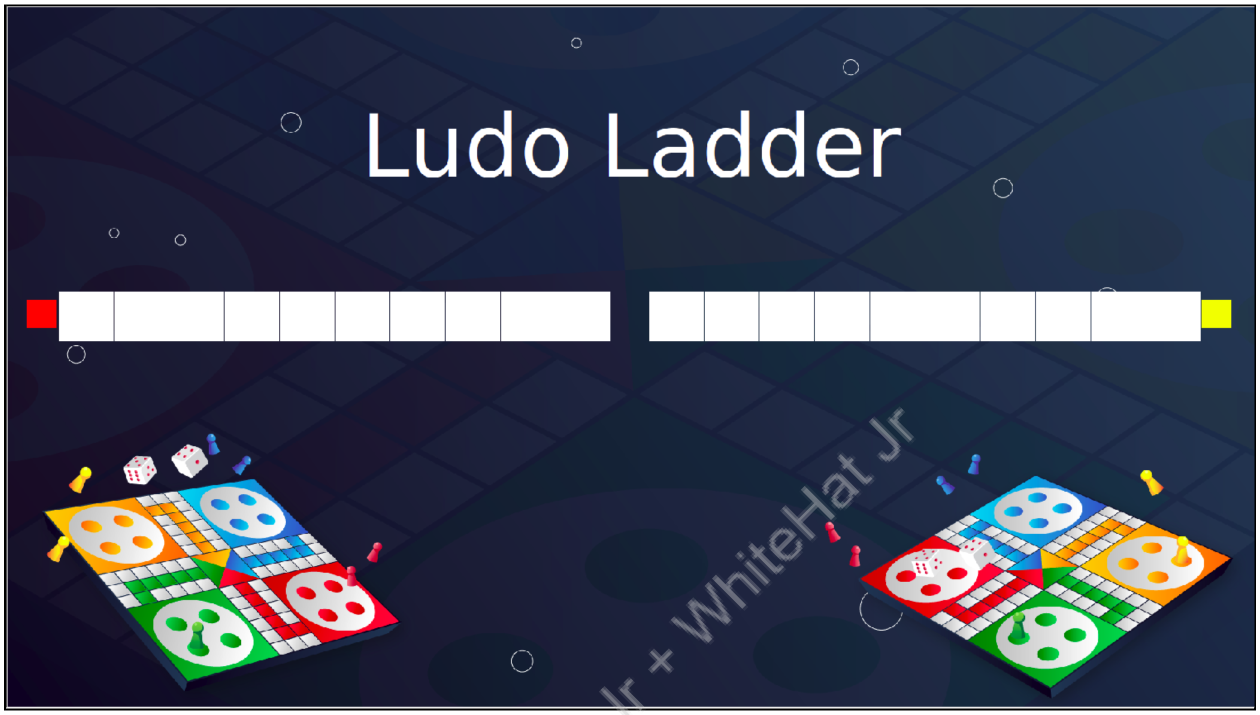


2. Write a function called as **leftBoard().** Using this function, create a board for the left player. Use loops to create 10 boxes from on the left side.

```
def leftBoard():
    global gameWindow
    global leftBoxes
    global screen_height

    xPos = 30
    for box in range(0,11):
        if(box == 0):
            boxLabel = Label(gameWindow, font=("Helvetica",30),
width=2, height=1, relief='ridge', borderwidth=0, bg="red")
            boxLabel.place(x=xPos, y=screen_height/2 - 88)
            leftBoxes.append(boxLabel)
            xPos +=50
        else:
            boxLabel = Label(gameWindow, font=("Helvetica",55),
width=2, height=1, relief='ridge', borderwidth=0, bg="white")
            boxLabel.place(x=xPos, y=screen_height/2- 100)
            leftBoxes.append(boxLabel)
            xPos +=75
```

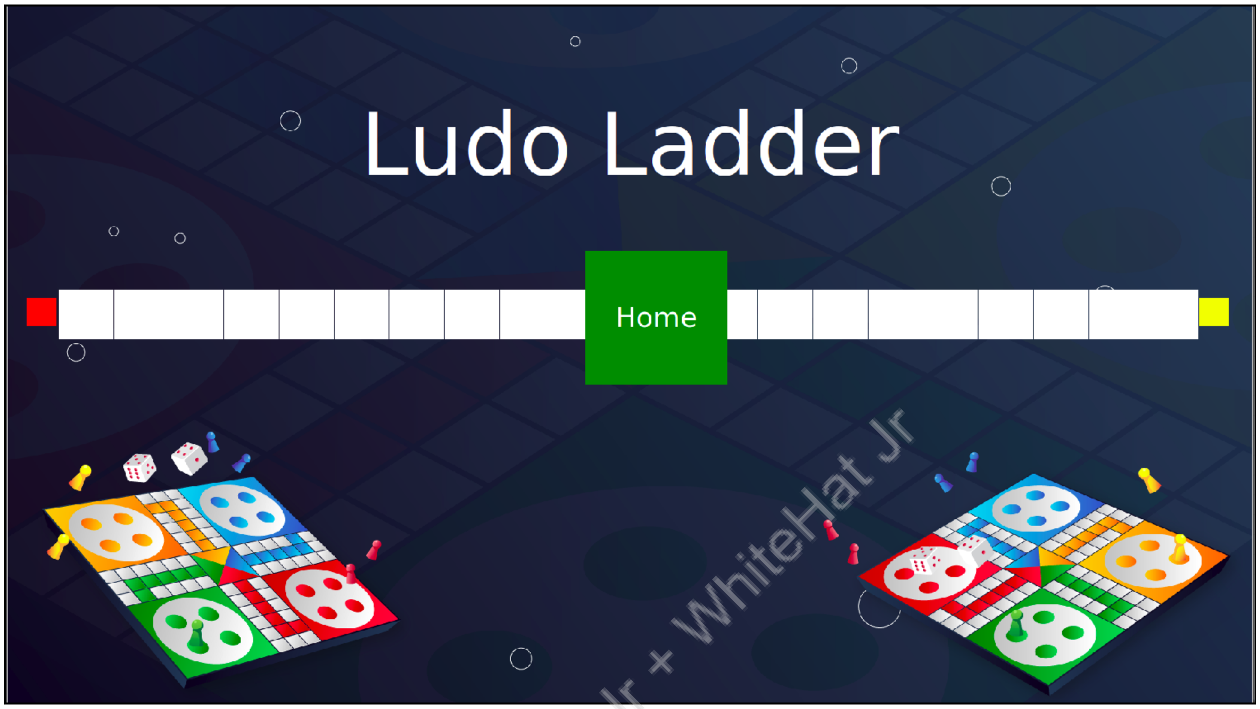
3. Do the same for the right player.

```
def rightBoard():

    global gameWindow

    global rightBoxes

    global screen_height


    xPos = 988

    for box in range(0,11):
        if(box == 10):

            boxLabel = Label(gameWindow, font=("Helvetica",30),
width=2, height=1, relief='ridge', borderwidth=0, bg="yellow")

            boxLabel.place(x=xPos, y=screen_height/2-88)

            rightBoxes.append(boxLabel)

            xPos +=50

        else:

            boxLabel = Label(gameWindow, font=("Helvetica",55),
width=2, height=1, relief='ridge', borderwidth=0, bg="white")

            boxLabel.place(x=xPos, y=screen_height/2 - 100)

            rightBoxes.append(boxLabel)

            xPos +=75
```



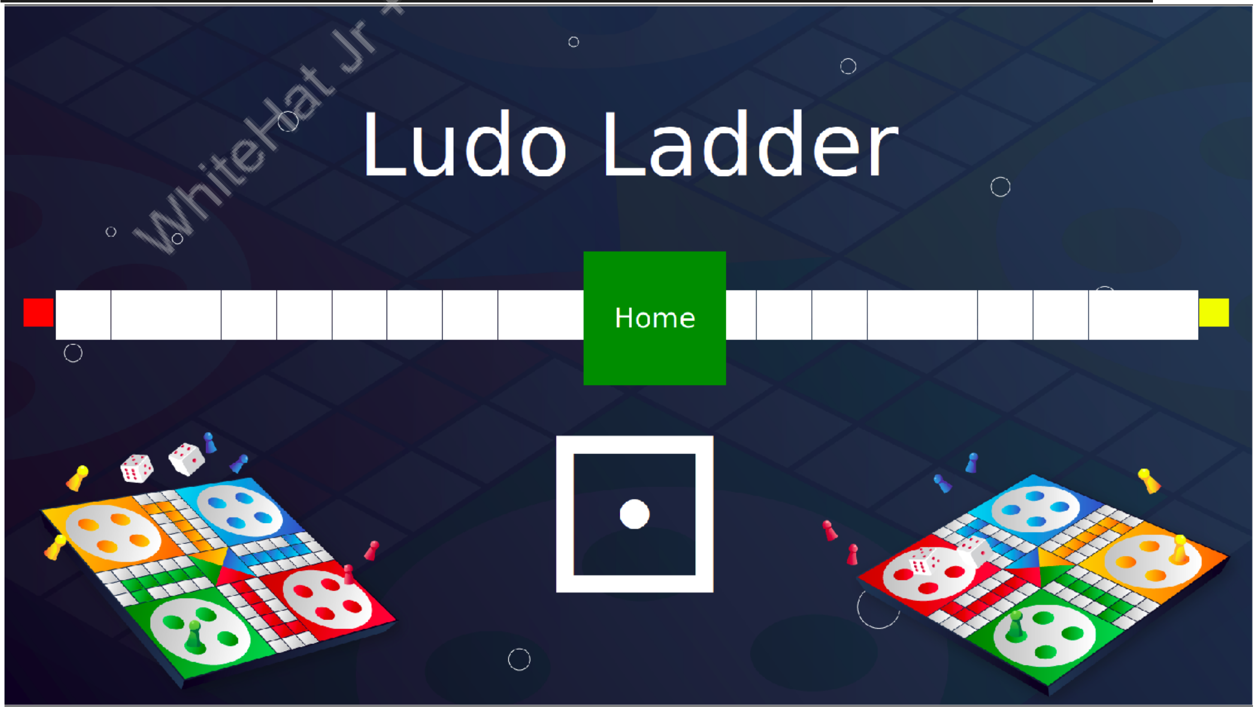
4. Write a finishingBox() , Inside this function create a box which will act as the final home for the player piece to reach.

```
def finishingBox():
   global gameWindow
   global finishingBox
   global screen_width
   global screen_height

   finishingBox = Label(gameWindow, text="Home", font=("Chalkboard
SE", 32), width=8, height=4, borderwidth=0, bg="green", fg="white")
   finishingBox.place(x=screen_width/2 - 68, y=screen_height/2 -160)
```



5. Add a dice for player to roll it to get a value to move the player pieces.

```
dice = canvas2.create_text(screen_width/2 + 10, screen_height/2 +
250, text = "\u2680", font=("Chalkboard SE",250), fill="white")
```



Till here you have created the game window GUI and added the player pieces for the game. Henceforth you'll add the functionality to the elements created.

6. Write a **rollDice()** function. In this function create a list of **diceChoices** . Using a random function, get one of the elements from the list. Also send these values to the server.

```
def rollDice():

   global SERVER

   #create a number variable in which the list of all the ASCII
characters of the string will be stored
   #Use backslash because unicode must have a backslash

diceChoices=['\u2680','\u2681','\u2682','\u2683','\u2684','\u2685']

   #configure the label
   value = random.choice(diceChoices)

   global playerType
   global rollButton
   global playerTurn

   rollButton.destroy()
   playerTurn = False

   if(playerType == 'player1'):
       SERVER.send(f'{value}player2Turn'.encode())

   if(playerType == 'player2'):
       SERVER.send(f'{value}player1Turn'.encode())
```

7. Create a rollButton. When this button is pressed then call the roll dice function.

```
   global rollButton
```

```
  rollButton = Button(gameWindow,text="Roll Dice", fg='black',
font=("Chalkboard SE", 15), bg="grey",command=rollDice, width=20,
height=5)
```

8. Write code to show this roll button to the player who has the turn to play.

```
global playerTurn
   global playerType
   global playerName


   if(playerType == 'player1' and playerTurn):
       rollButton.place(x=screen_width / 2 - 80, y=screen_height/2
+ 250)
   else:
       rollButton.pack_forget()
```

9. Write a **handleClient()** function which takes two parameters **player_socket and player_name** . In the **playerType** variable get the player type from the **CLIENTS** dictionary.

   If it's player1 turn set the key **turn** to True.

   And send the message that it's player1 turn.

   Else set the key **turn** to False and send the message that it's player2 turn.

```
def handleClient(player_socket,player_name):
   global CLIENTS

   # Sending Initial message
   playerType =CLIENTS[player_name]["player_type"]
   if(playerType== 'player1'):
       CLIENTS[player_name]['turn'] = True
       player_socket.send(str({'player_type' :
CLIENTS[player_name]["player_type"] , 'turn':
CLIENTS[player_name]['turn'], 'player_name' : player_name
}).encode())
   else:
       CLIENTS[player_name]['turn'] = False
       player_socket.send(str({'player_type' :
CLIENTS[player_name]["player_type"] , 'turn':
CLIENTS[player_name]['turn'] }).encode())



   while True:
       try:
           message = player_socket.recv(2048)
           if(message):
               for cName in CLIENTS:
                   cSocket = CLIENTS[cName]["player_socket"]
                   cSocket.send(message)
       except:
           pass
```

What's NEXT?

In the next class, we will be working on finishing the game and adding restart button to play again.

Expand Your Knowledge:

Explore more about the creating GUI using Tkinter through this link :

https://realpython.com/python-gui-tkinter/