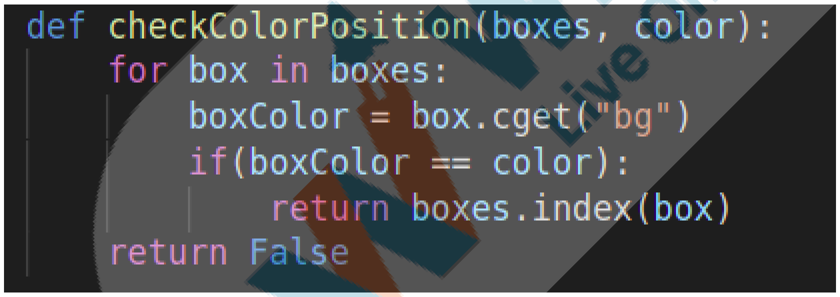
Topic	**GAME MECHANICS-I**	
Class Description	**Students will learn to display the player piece by color on the board. The student will also learn to to add logic for the player piece movement.**	
Class	**PRO C206**	
Class time	**45 mins**	
Goal	● Display the player piece by color. ● Write logic for player piece movement	
Resources Required	● Teacher Resources: ○ Laptop with internet connectivity ○ Earphones with mic ○ Notebook and pen ○ Smartphone ● Student Resources: ○ Laptop with internet connectivity ○ Earphones with mic ○ Notebook and pen	
Class structure	**Warm-Up** **Teacher - led Activity 1** **Student - led Activity 1** **Wrap-Up**	**10 mins** **10 mins** **20 mins** **5 mins**

WARM-UP SESSION - 10 mins

Teacher Action	**Student Action**
Hey <student's name>. How are you? It's great to see you! Are you excited to learn something new today?	**ESR**: Hi, thanks, Yes I am excited about it!

Can you recall what we did in the last class? In the last class we created the game window and added the left board and the right board to the window. We also added the dice which will be rolled for the players to play. In today's class we'll write code to get the player positions and their movements in the game board.	*Student recalls the concepts covered in last class*
Q&A Session	
Question	**Answer**
TEACHER-LED ACTIVITY - 10 mins	
Teacher Initiates Screen Share	
ACTIVITY ● ● **Adding left and right boxes for the two players.**	
Teacher Action	**Student Action**
Teacher downloads the boilerplate code from Teacher Activity 1 *Note :- code for **checkColorPosition()** function is provided in the boilerplate code.*	**ESR:** **Yes!**
We'll have the dice and the board ready but we haven't decided how the players will move and how we will know that the player has won.	

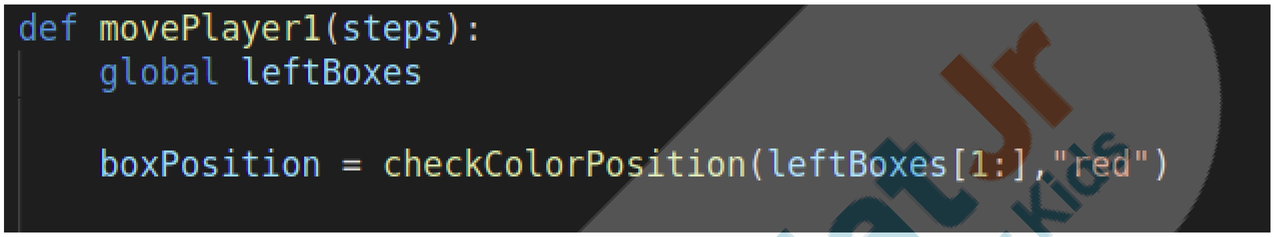
Can you tell me how can we do that? So for that we'll first start by getting the position of the player pieces. We'll write a **checkColorPosition()** function. This function will take two parameters: first the boxes and second the color. Inside the function we'll get all the boxes from the boxes array which we created while creating the board. Using the **cget()** function we'll get the background color of the box and store it in the **boxColor** variable. Then using an If condition we'll check if the box color matches with our color and then return the box index. Else return False.	**ESR:** We are using red and yellow colored pieces to indicate the positions of the players. So each time a dice is rolled we'll move the pieces .
Note:- checkColorPosition code is present as boilerplate. 	
Now we can get the position of the player piece. So let's write the logic for the movement of player 1's piece. We know that player 1 will always be on the left side of the screen. So let's start by defining the function. We'll call this function **movePlayer1()**. This function will take a parameter which is the steps.	**ESR:** import/ use all the necessary variables in the function.

```
def checkColorPosition(boxes, color):
    for box in boxes:
        boxColor = box.cget("bg")
        if(boxColor == color):
            return boxes.index(box)
    return False
```

Inside the function :-

- Call the global variable **leftBoxes.**
- Using the **checkColorPosition()** function, get the position of the player piece and store it in the **boxPosition** variable.

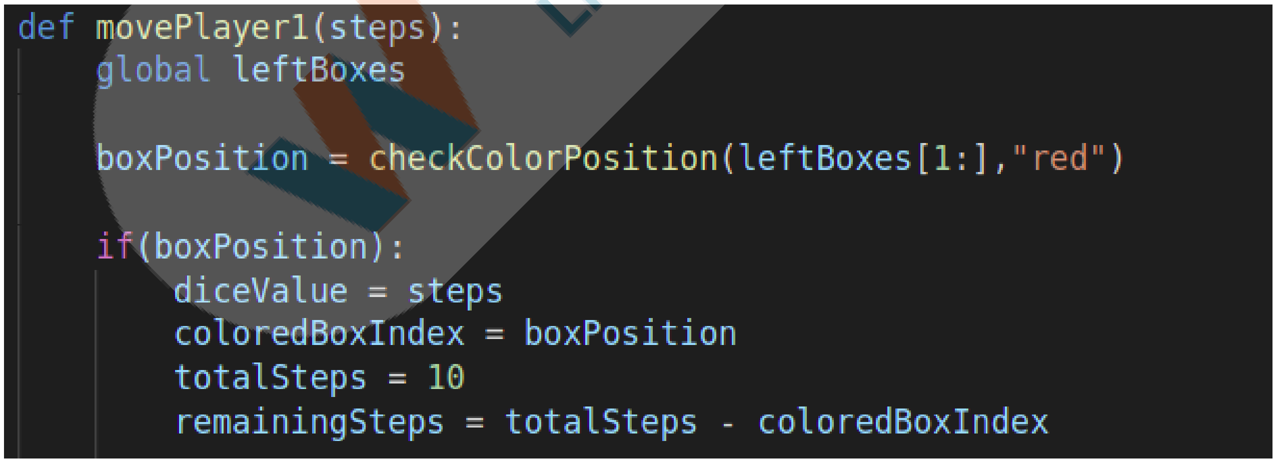
In the client.py file



```
def movePlayer1(steps):
    global leftBoxes

    boxPosition = checkColorPosition(leftBoxes[1:],"red")
```

Now using the If condition we'll check if there is a value inside the **boxPosition** variable.
Inside the if condition.

- Declare a **diceValue** and set its value as **steps.**
- Declare the **colouredBoxIndex** variable and set its value as **boxPosition.**
- Declare a variable called as **steps** and set 10 as its value
- Declare the **remainingSteps** variable and have **totalSteps -coloredBoxIndex** as it's value.



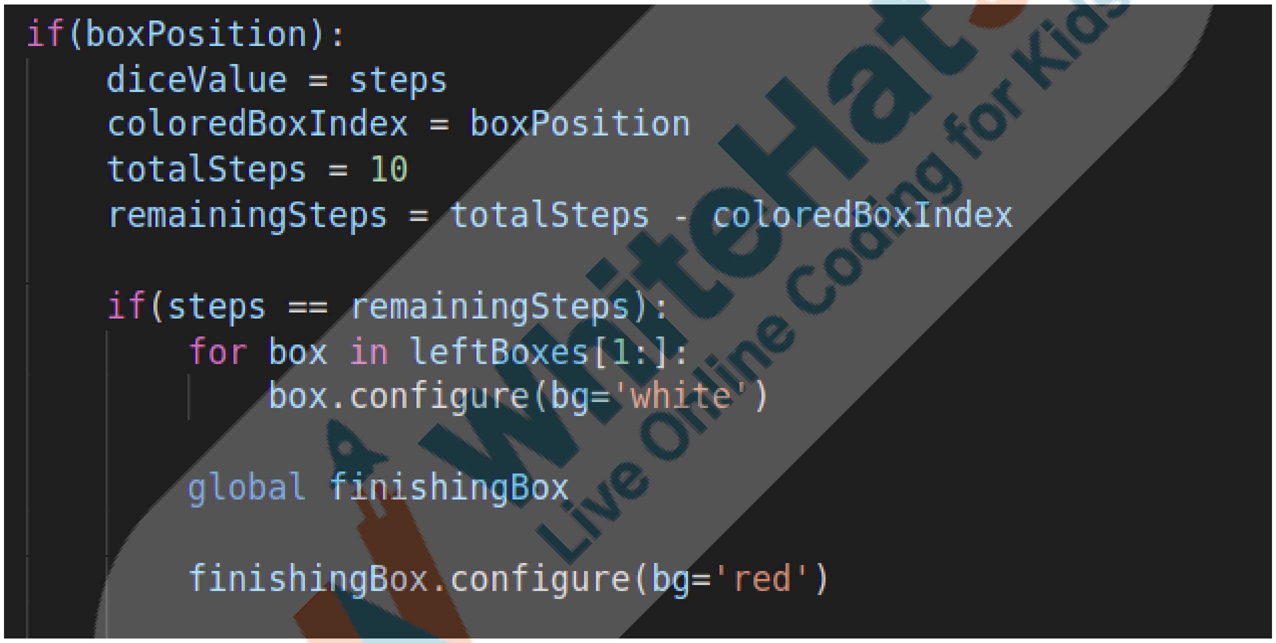
```
def movePlayer1(steps):
    global leftBoxes

    boxPosition = checkColorPosition(leftBoxes[1:],"red")

    if(boxPosition):
        diceValue = steps
        coloredBoxIndex = boxPosition
        totalSteps = 10
        remainingSteps = totalSteps - coloredBoxIndex
```

Inside the first If condition write the other if condition where

we'll check if the **steps** that are passed to the function are equal to the **remainingSteps** .
Using a for loop get all the boxes inside the **leftBoxes** array and using **configure()** method set the background color to white.

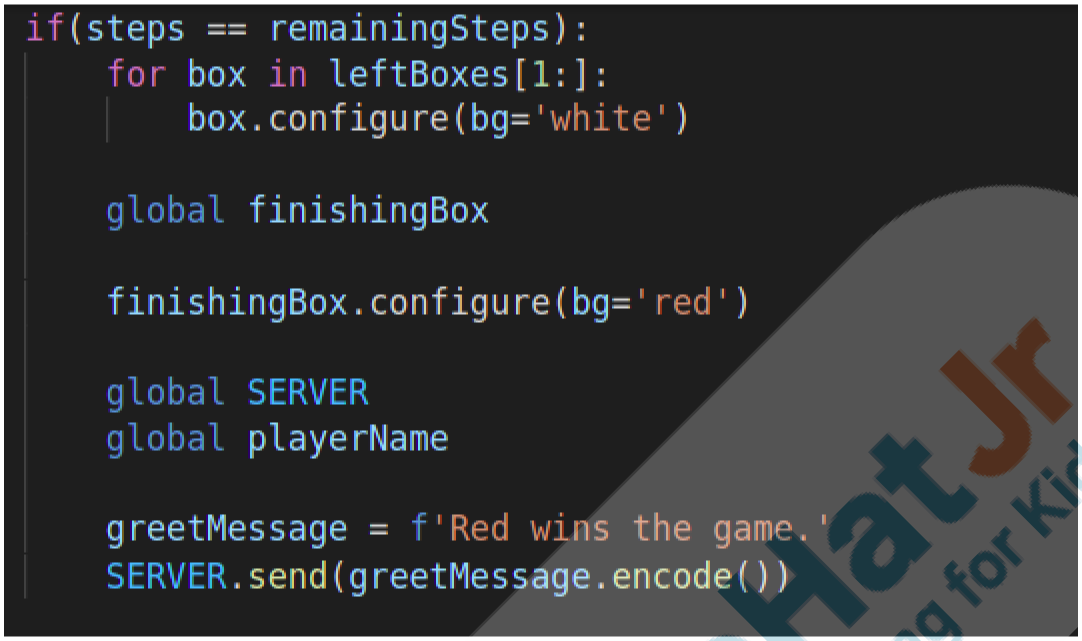
If the player 1 wins then we'll set the finishing box to red color. To do so :-

- Use the global **finishingBox** variable .
- Using the **configure()** method set the background to red for the finishing box.



```
if(boxPosition):
    diceValue = steps
    coloredBoxIndex = boxPosition
    totalSteps = 10
    remainingSteps = totalSteps - coloredBoxIndex

    if(steps == remainingSteps):
        for box in leftBoxes[1:]:
            box.configure(bg='white')

        global finishingBox

        finishingBox.configure(bg='red')
```

Then use the global **SERVER and playerName** variables.
Declare a **greetMessage** variable which will have a text "Red wins the game".
Using the **SERVER.send()** send the message to the server.



```
if(steps == remainingSteps):
    for box in leftBoxes[1:]:
        box.configure(bg='white')

    global finishingBox

    finishingBox.configure(bg='red')

    global SERVER
    global playerName

    greetMessage = f'Red wins the game.'
    SERVER.send(greetMessage.encode())
```

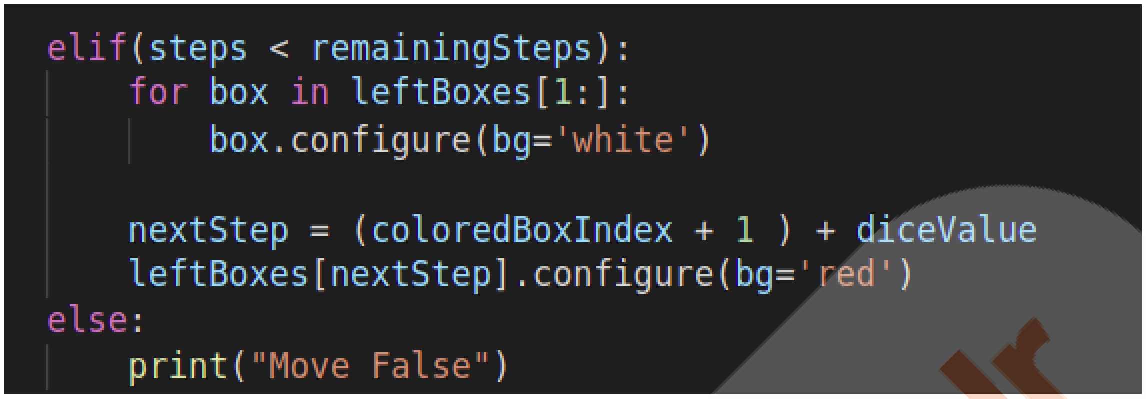
So now we have written code for the condition that if player 1 has won , there is now another scenario where the player still hasn't won yet .

For this scenario we'll write an elif condition to check if the number of **steps** are lesser than the **remainingSteps.**

If they are then we'll set the boxes to white color using the **configure()** .

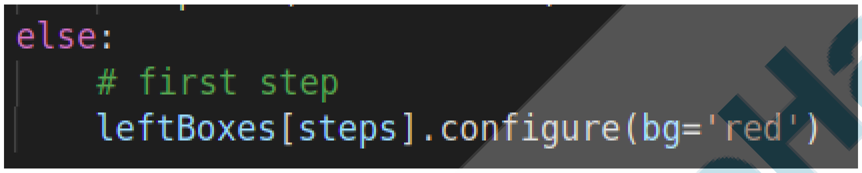
To set the position of the player on the board
Declare the **nextStep** variable and store **colorBoxIndex** and **dice** value in it.
Using **configure()** method set the box to red color.

Else print that it's a false statement.



```
elif(steps < remainingSteps):
    for box in leftBoxes[1:]:
        box.configure(bg='white')

    nextStep = (coloredBoxIndex + 1 ) + diceValue
    leftBoxes[nextStep].configure(bg='red')
else:
    print("Move False")
```

Else set the box with the index number of steps to red color.



```
else:
    # first step
    leftBoxes[steps].configure(bg='red')
```

Till here we have written code to get the colored position of player 1. Similarly can you do the same for player2? Let's get started then.	**ESR:** Yes

Teacher Stops Screen Share

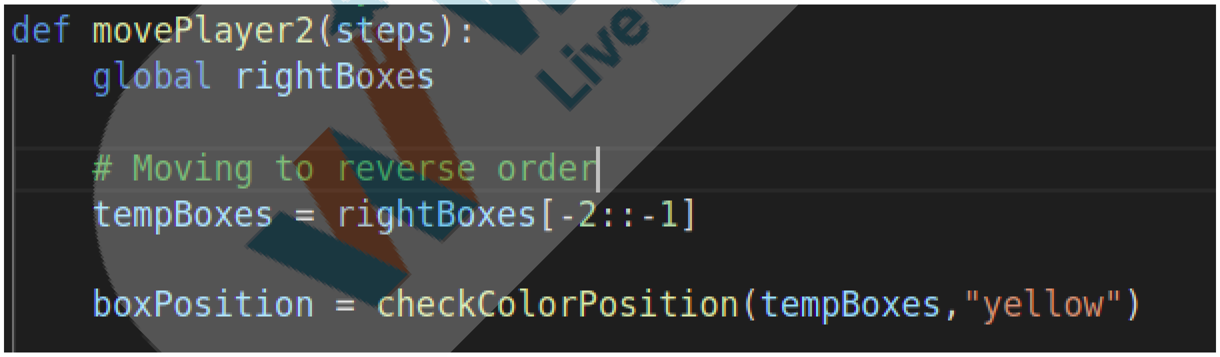
STUDENT-LED ACTIVITY - 20 mins

- **Ask the student to press the ESC key to come back to the panel.**
- **Guide the student to start Screen Share.**
- **The teacher gets into Full screen.**

ACTIVITY

- **Write code for different scenarios of the player 2.**
- **Code to decide the player turn.**

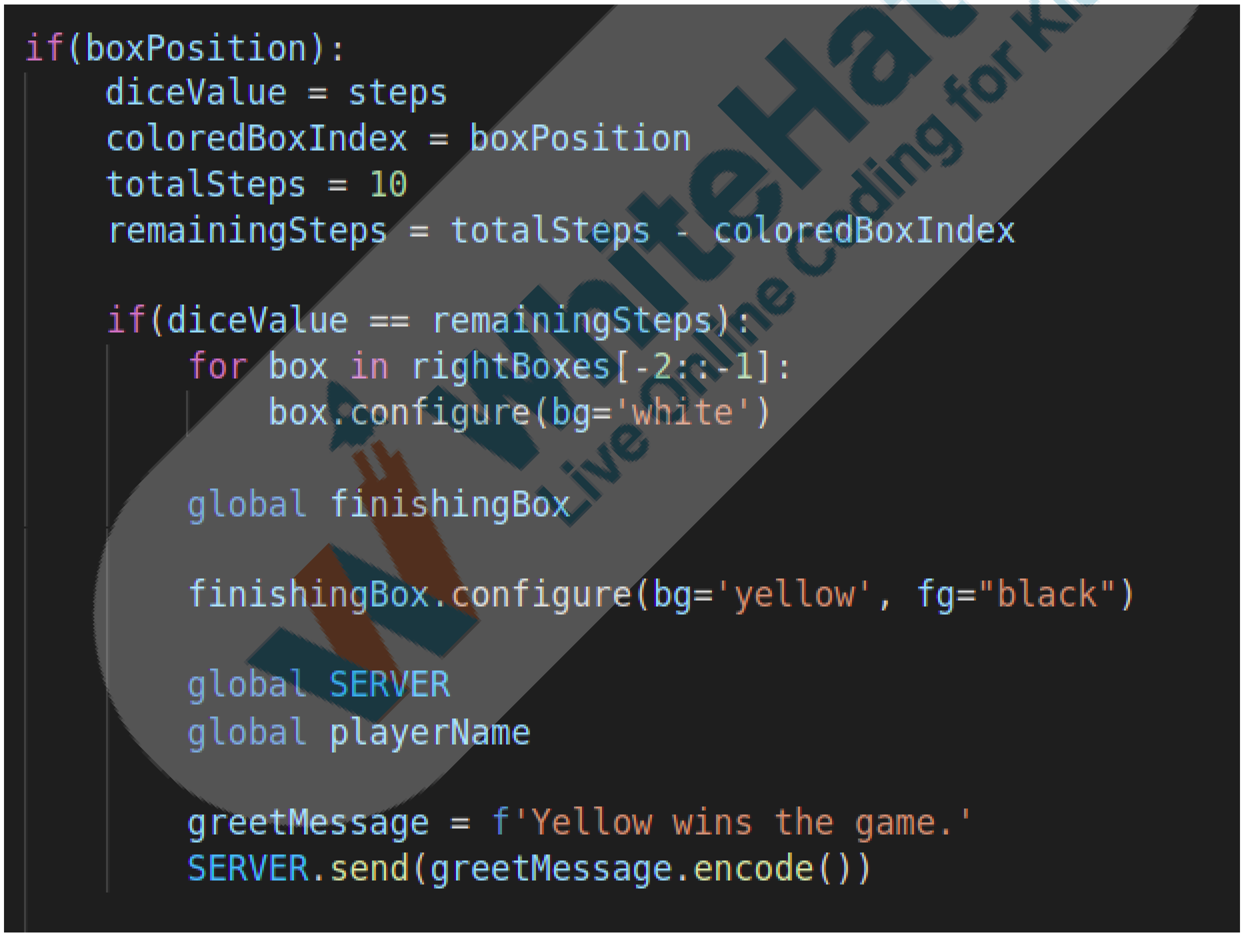
Teacher Actions	Student Action
The teacher guides the student to clone the code from [Student Activity 1](#) Alright so the code for player 2 will be the same as we have for player 1 but there will be little difference . Can you tell me what that difference is? Yes! The boxes in the player 2 side will be reversed. Let's start by defining the **movePlayer2()** function. Define the **movePlayer2()** function and Use the global **rightBoxes** variable. Create a variable named **tempBoxes** and using the array iteration method get the boxes in reverse order. [-2::-1] is like running the loop in reverse to get the elements of an array. Call the **checkColorPosition()** function and pass the tempBoxes and yellow color as the parameters.	*Student clones the code from [Student Activity 1](#)* **ESR:** The difference is the position of the boxes for player 2. *Student starts writing code for **movePlayer2()***



```
def movePlayer2(steps):
    global rightBoxes

    # Moving to reverse order
    tempBoxes = rightBoxes[-2::-1]

    boxPosition = checkColorPosition(tempBoxes,"yellow")
```

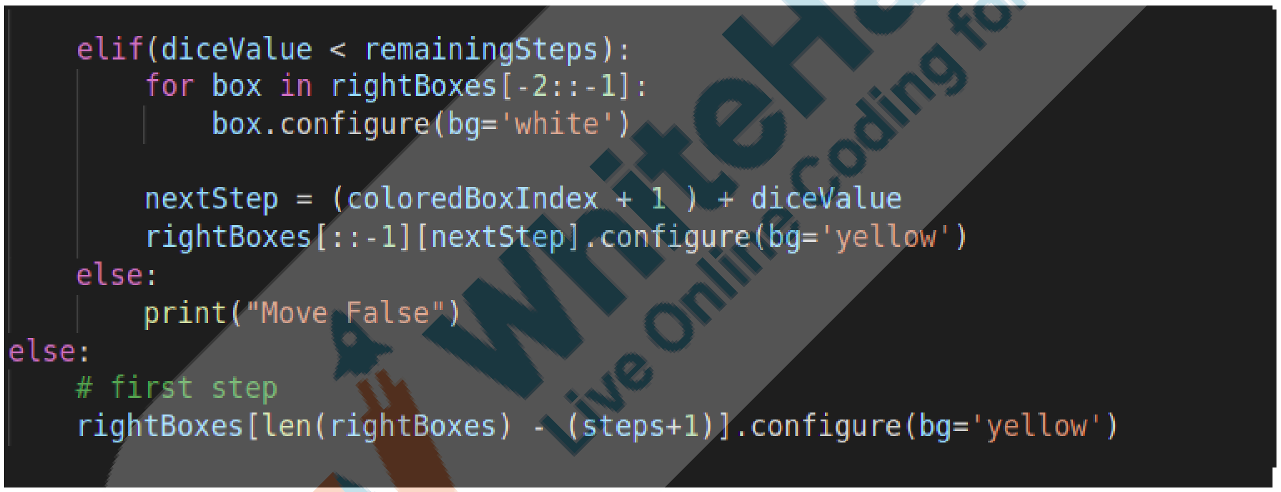
Using if condition check if the **checkColorPosition()** function returns the position then Declare **diceValue, coloredBoxIndex, totalSteps and remainingSteps** variables and assign their respective	player.

values. Using if condition check if the **diceValue** is equal to the **remainingSteps.** Then using for loop on the rightBoxes array set the background color to white using **configure()** function. Also use the global **SERVER** variable. And send the greeting message saying “Yellow wins the game”	



```
if(boxPosition):
    diceValue = steps
    coloredBoxIndex = boxPosition
    totalSteps = 10
    remainingSteps = totalSteps - coloredBoxIndex

    if(diceValue == remainingSteps):
        for box in rightBoxes[-2::-1]:
            box.configure(bg='white')

        global finishingBox

        finishingBox.configure(bg='yellow', fg="black")

        global SERVER
        global playerName

        greetMessage = f'Yellow wins the game.'
        SERVER.send(greetMessage.encode())
```

If the remaining steps for the player to win are more than the dice value then : Use a for loop to get all the boxes and using **configure()** set the background to white. In the **nextStep** variable get the **coloredBoxIndex + diceValue** value. Set the background yellow for the box at this particular value.	*Student Completes the code for movePlayer2().* .



```
    elif(diceValue < remainingSteps):
        for box in rightBoxes[-2::-1]:
            box.configure(bg='white')

        nextStep = (coloredBoxIndex + 1 ) + diceValue
        rightBoxes[::-1][nextStep].configure(bg='yellow')
    else:
        print("Move False")
else:
    # first step
    rightBoxes[len(rightBoxes) - (steps+1)].configure(bg='yellow')
```

Awesome job , now we know how each of our players will move. So till now we have been sending messages from functions such as **rollDice()** , so next time we'll process these messages to get the desired outcome.	
WRAP-UP SESSION - 5 mins	
Quiz time - Click on In-Class Quiz	
Question	**Answer**

End the quiz panel	
FEEDBACK ● **Appreciate the student for his/her efforts in the class.** ● **Ask the student to make notes for the reflection journal along with the code they wrote in today's class.**	
Teacher Action	**Student Action**
You get hats-off for your excellent work! In the next class, we'll write functions to send the message from client to client.	*Make sure you have given at least 2 hats-off during the class for:* Creatively Solved Activities +10 Great Question +10 Strong Concentration +10
Project Discussion	
Teacher Clicks ✖ End Class	
ADDITIONAL ACTIVITIES	

Additional Activities *Encourage the student to write reflection notes in their reflection journal using markdown.* Use these as guiding questions: ● What happened today? ○ Describe what happened. ○ The code I wrote. ● How did I feel after the class? ● What have I learned about programming and developing games? ● What aspects of the class helped me? What did I find difficult?	*The student uses the markdown editor to write her/his reflections in the reflective journal.*

ACTIVITY LINKS		
Activity Name	**Description**	**Link**
Teacher Activity 1	Boilerplate code	https://github.com/pro-whitehatjr/PRO-206-TA
Teacher Activity 2	Reference code	https://github.com/pro-whitehatjr/C206-reference-code
Student Activity 1	Boilerplate Code	https://github.com/pro-whitehatjr/PRO-206-SA