Topic	GAME MECHANICS-II	
Class Description	**Students will finish the ludo ladder game. Students will learn to display the player joining message as the player joins. Also, display the turn of the player.**	
Class	**PRO C207**	
Class time	**45 mins**	
Goal	• Display the player joining message • Add functionality and display the score.	

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? Can you recall what we did in the last class?	**ESR**: Hi, thanks, Yes I am excited about it! *The student recalls the*

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 players positions and their movements on the game board.	*concepts covered in the 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 :- Declaring winning message, creating reset button , **updateScore()** , **handleResetGame()** function is provided in the boilerplate code.* *These functions are explained where they are required*	
Awesome job! Earlier we had sent some messages to the server in a function such as **rollDice().** So these messages need to be received somewhere and processed to see the desired outputs. Such as which player has the turn to play or what is the dice value when it is rolled.	

*Teacher starts writing the code for **receivedMsg()** function.*

We have a **receivedMsg()** function which does this work.
In this function get all the required global variables.

To keep the server listening continuously use the **while True** condition.

Declare a variable called as **message**, in this variable receive all the messages using **SERVER.recv** and decode them.

```
def recivedMsg():
    global SERVER
    global playerType
    global playerTurn
    global rollButton
    global screen_width
    global screen_height
    global canvas2
    global dice
    global gameWindow
    global player1Name
    global player2Name
    global player1Label
    global player2Label
    global winingFunctionCall


    while True:
        message = SERVER.recv(2048).decode()
```

Using the if condition check if there is a “player_type’ key in this message.
Then using the eval**()** function convert the message to the dict format.
Get the player type from the message and store in **playerType** variable, and the player turn in **playerTurn** variable.

If the message contains player names do the similar for player names, Store the player1 name in **player1Name** and player 2 Name in **player2Name**	
	
Else, if there is one of the values of dice in the message then using the **itemConfigure()** method displays the message on the screen.	

```
elif('⚀' in message):
    # Dice with value 1
    canvas2.itemconfigure(dice, text='\u2680')
elif('⚁' in message):
    # Dice with value 2
    canvas2.itemconfigure(dice, text='\u2681')
elif('⚂' in message):
    # Dice with value 3
    canvas2.itemconfigure(dice, text='\u2682')
elif('⚃' in message):
    # Dice with value 4
    canvas2.itemconfigure(dice, text='\u2683')
elif('⚄' in message):
    # Dice with value 5
    canvas2.itemconfigure(dice, text='\u2684')
elif('⚅' in message):
    # Dice with value 6
    canvas2.itemconfigure(dice, text='\u2685')
```

If we have the "**win the game**" in the message and the **winingFunctionCall** is 0. Then increment the **WiningFunctionalCall** variable by 1 and call the **handleWin()** function.
handleWin() function will destroy the roll buttons from both the player screen and set the reset button on the screen.

```
elif('wins the game.' in message and winingFunctionCall == 0):
    winingFunctionCall +=1
    handleWin(message)
```

Here we are calling the sudo handleWin function. Can you write code for **handleWin()** function to reset the player position and display the dice again on the screen?

Let's get you 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 Fullscreen.**	
ACTIVITY • **Code to handle the process after a player wins.**	
Teacher Actions	**Student Action**
Teacher guides student to download code from Student Activity 1	*student to download code from Student Activity 1*
In the **handleWin()** function get all the global variables such as **playerType, rollButton, canvas2 , winingMessage , screen_width, screen_height, resetButton.** Then check if the message contains the word “Red”. If it does then check if the playerType is player2. Then destroy the rollButton. Similarly if the message has the word “Yellow” then check if the playerType is player1. And remove the rollButton using the **destroy()** method. Using the split() method split the message to get the desired message and display it on the screen using **itemconfigure() method.** Place the reset button on the screen.	

```
def handleWin(message):
    global playerType
    global rollButton
    global canvas2
    global winingMessage
    global screen_width
    global screen_height
    global resetButton

    #destroying button
    if('Red' in message):
        if(playerType == 'player2'):
            rollButton.destroy()

    if('Yellow' in message):
        if(playerType == 'player1'):
            rollButton.destroy()

    # Adding Wining Message
    message = message.split(".")[0] + "."
    canvas2.itemconfigure(winingMessage, text = message)

    #Placing Reset Button
    resetButton.place(x=screen_width / 2 - 80, y=screen_height - 220)
```

We have the dice ready but we don't know whose turn it will be to start the game . What can we do to get whose turn it is? Yes! Let's do that.	**ESR:** In the message that we are sending from client to server we are also sending the player's turn. So we can use it to get which player has his/her turn. *Student codes to get the player turn from the*

Write a if condition to check if the **player1Turn** or **player2Turn** in the message. Inside the condition declare the **diceChoices** variable. Set an array with all the dice sides as value to **diceChoices.** If it's player2Turn in message then call the **movePlayer1()** and pass **diceValue** as its parameter. If it's player1Turn in message then call the **movePlayer2()** and pass **diceValue** as its parameter.	*message.* *If it's player2Turn then call the movePlayer1() function.* *If it's player1Turn then call the movePlayer2() function.*

```
if('player1Turn' in message or 'player2Turn' in message):
    diceChoices=['⚀','⚁','⚂','⚃','⚄','⚅']
    diceValue = diceChoices.index(message[0]) + 1

    if('player2Turn' in message):
        movePlayer1(diceValue)


    if('player1Turn' in message):
        movePlayer2(diceValue)
```

Now we can know which player will have its turn so now we just need to show the roll dice button to the player who has the turn. What can we do to create the roll button? Write a if condition to check if it's player1Turn and playerType is player1 in the message. In the **command** call the **rollDice** function.	**ESR:** It will be the same as we decided which player will have his/her turn. The message that we are receiving has the player turn and player type .

Write another condition using **elif** to check for player2Turn and playerType is **player2.** In the **command** call the **rollDice** function.	*Student codes to check the player turn and player type to*

```
if('player1Turn' in message and playerType == 'player1'):
            playerTurn = True
            rollButton = Button(gameWindow,text="Roll Dice", fg='black',
font=("Chalkboard SE", 15), bg="grey",command=rollDice, width=20, height=5)
            rollButton.place(x=screen_width / 2 - 80, y=screen_height/2  +
250)

        elif('player2Turn' in message and playerType == 'player2'):
            playerTurn = True
            rollButton = Button(gameWindow,text="Roll Dice", fg='black',
font=("Chalkboard SE", 15), bg="grey",command=rollDice, width=20, height=5)
            rollButton.place(x=screen_width / 2 - 80, y=screen_height/2  +
260)
```

Teacher Guides Student to Stop Screen Share

WRAP-UP SESSION - 5 mins

Quiz time - Click on In-Class Quiz

Question	Answer

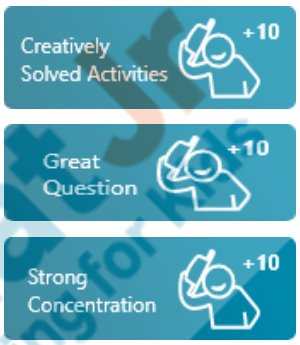
End the quiz panel

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**

<table>
<tr><td colspan="2">they wrote in today's class.</td></tr>
<tr><td>Teacher Action</td><td>Student Action</td></tr>
<tr><td>You get hats-off for your excellent work!

In the next class, we'll write functions to send the message from client to client.</td><td>Make sure you have given at least 2 hats-off during the class for:

 
</td></tr>
<tr><td>Project Discussion</td><td></td></tr>
<tr><td colspan="2">Teacher Clicks 
</td></tr>
<tr><td colspan="2">ADDITIONAL ACTIVITIES</td></tr>
<tr><td>Additional Activities

How can we make the game more interesting?

Yes, It would be interesting to know against whom we are playing the game

First let's start by creating a name board to display the name on the game screen.</td><td>ESR:
We can display the name of the players on the screen along with the score .</td></tr>
</table>

Using the **create_text()** method shows the name of the player 1.
Do the same for player 2.

```
   # Creating name board
   player1Label = canvas2.create_text(400,  screen_height/2 + 100, text =
player1Name, font=("Chalkboard SE",80), fill='#fff176' )
   player2Label = canvas2.create_text(screen_width - 300, screen_height/2 +
100, text = player2Name, font=("Chalkboard SE",80), fill='#fff176' )
```

Alright now let's add the scoreboard to see the score of the players.

Using the **create_text()** method add the score to player1
Do the same for player 2.

```
 # Creating Score Board
   player1ScoreLabel = canvas2.create_text(400,  screen_height/2 - 160,
text = player1Score, font=("Chalkboard SE",80), fill='#fff176' )
   player2ScoreLabel = canvas2.create_text(screen_width - 300,
screen_height/2 - 160, text = player2Score, font=("Chalkboard SE",80),
fill='#fff176' )
```

ACTIVITY LINKS		
Activity Name	**Description**	**Link**
Teacher Activity 1	Boilerplate code	https://github.com/pro-whitehatjr/PRO-207-TA
Teacher Activity 2	Reference code	https://github.com/pro-whitehatjr/PRO-C207-Reference-code
Student Activity 1	Boilerplate Code	https://github.com/pro-whitehatjr/PRO-207-SA