Milestone Project Help

Section 7, Lecture 48

Tic Tac Toe Project Helpful Hints

In this text lecture we will just have a useful guide for helping you complete the project! Sometimes this project can feel overwhelming, like being thrown into the deep end of the pool and told "Now learn to swim!". So to help out with this, here's a guide to help you begin in the right direction! (Note, there's lots of ways to accomplish this task, so your code may look completely different than the given solution). If the hints below still aren't enough, check out the "Walkthrough Workbook" notebook for even more help!

- **First off, make sure you understand the project scope. What needs to happen?**

    1. We need to print a board.
    2. Take in player input.
    3. Place their input on the board.
    4. Check if the game is won,tied, lost, or ongoing.
    5. Repeat c and d until the game has been won or tied.
    6. Ask if players want to play again.

- **Ok so we got a general idea of what we need to do. Let's break it down by steps. If you're having trouble with the project, come and check this lecture if you ever get stuck.**

    PROJECT HINTS:

    1. Start by deciding how you will store the player's marker positions (Xs and Os). Let's choose a list, where each index corresponds with a number on a keypad, which in turn corresponds with a position on the 3 by 3 board.
    2. Create a list called **board** which will keep track of the player markers.
        a. The list should already be the same length as your intended board.
    3. Create a function that will print a board. Not just the list, but an actual representation of a board! This can be done with multiple print statements within the function. Think about how you would take elements from the list and print them out into the board. (You can also clear output in jupyter notebook with **clear_output()** . You need to import this, so at the top of a cell copy and paste: **from IPython.display import clear_output**

4. Write a function which takes an input marker string 'X' or "O' and a given number and stores it to a list at that appendix.

. You might have to look up how to take input in python! **input()**

    a. Play around with **input()** to make sure you understand it

5. Write a function that takes in the board and a player marker and checks it theres a win or a tie.

. The checking for a win should be a series of a bunch checks, for example: **(board[7] == board[8] == board[9] == marker)**would check the first top row if they all match a player's marker.

6. Check for a tie (this means nobody won and the board list is full!)

7. Now begin writing functions that begin game play.

8. You'll need to write a function which starts combining and calling the different functions you've made within it.

. For example, a function which asks for a player's move, then updates the board,then checks for a win, then prints out the board.

9. How can you keep the game continually going?

. Maybe try using a **while** loop.

    a. Something like, while the board isn't full and nobody's won...

10. You might want to think about how to use boolean objects as markers of the game's status.

11. Alright you should have enough now to begin piecing things together!

12. If you're still stuck, try googling around for "Python Tic Tac Toe" for ideas of how to put all your pieces and functions together!

Great job and remember, this is a milestone project, so its meant to be challenging!

Other things that you may want to implement:

1. Taking in a player's input:

If you're confused on how to take in a players input, you have to use **input()** for Python 3, or **raw_input()** for Python 2.. You can use it in the following manner:

board_position = input("Please enter a number for the board position choice: ")

Then you would see a pop-up occur with a text box asking you for your input. Type in the number and press enter, now the variable **board_position** is an integer you can use for your code. For more on the differences between raw_input() and input(), check out this answer.

2. Checking an input and asking again:

In my solution I take into account if a player possibly puts in an incorrect input (like passing a string 'hello' when asked for their board position choice). You can take care of this by using a **while loop!**