Revisiting the Tic-Tac-Toe Program

by Sophia

WHAT'S COVERED

In this lesson, you will expand on the Tic-Tac-Toe program to make use of loops. Specifically, this lesson covers:

1. Tic-Tac-Toe With Loops
2. Adding Validation

1. Tic-Tac-Toe With Loops

This is a great time to revisit the Tic-Tac-Toe game that we worked on at the end of the last challenge, as we had many instances where we could make use of loops. Let’s first look at the code that we ended with.

TRY IT

Directions: If you don’t have an existing file in the IDE, let’s enter in the following code that we originally ended with the last time we visited the Tic-Tac-Toe program.

```
board = [ ["-", "-", "-"],
          ["-", "-", "-"],
          ["-", "-", "-"] ]

print(board[0])
print(board[1])
print(board[2])

#X first move
col = int(input("X player, select a column 1-3: "))
row = int(input("X player, select a row 1-3: "))
col -= 1
row -= 1
```

```
board[row][col] = "X"
print(board[0])
print(board[1])
print(board[2])

#O first move
col = int(input("O player, select a column 1-3: "))
row = int(input("O player, select a row 1-3: "))
col -= 1
row -= 1

if board[row][col] == '-':
  board[row][col] = "O"
else:
  print("Oops, that spot was already taken. ")
print(board[0])
print(board[1])
print(board[2])

#X second move
col = int(input("X player, select a column 1-3: "))
row = int(input("X player, select a row 1-3: "))
col -= 1
row -= 1

if board[row][col] == '-':
  board[row][col] = "X"
else:
  print("Oops, that spot was already taken. ")
print(board[0])
print(board[1])
print(board[2])

#O second move
col = int(input("O player, select a column 1-3: "))
row = int(input("O player, select a row 1-3: "))
col -= 1
row -= 1

if board[row][col] == '-':
  board[row][col] = "O"
```

```
else:
  print("Oops, that spot was already taken. ")
print(board[0])
print(board[1])
print(board[2])

#X third move
col = int(input("X player, select a column 1-3: "))
row = int(input("X player, select a row 1-3: "))
col -= 1
row -= 1

if board[row][col] == '-':
  board[row][col] = "X"
else:
  print("Oops, that spot was already taken. ")
print(board[0])
print(board[1])
print(board[2])

#0 third move
col = int(input("O player, select a column 1-3: "))
row = int(input("O player, select a row 1-3: "))
col -= 1
row -= 1

if board[row][col] == '-':
  board[row][col] = "O"
else:
  print("Oops, that spot was already taken. ")
print(board[0])
print(board[1])
print(board[2])

#X fourth move
col = int(input("X player, select a column 1-3: "))
row = int(input("X player, select a row 1-3: "))
col -= 1
row -= 1

if board[row][col] == '-':
  board[row][col] = "X"
```

```
else:
  print("Oops, that spot was already taken. ")
print(board[0])
print(board[1])
print(board[2])

#O fourth move
col = int(input("O player, select a column 1-3: "))
row = int(input("O player, select a row 1-3: "))
col -= 1
row -= 1

if board[row][col] == '-':
  board[row][col] = "O"
else:
  print("Oops, that spot was already taken. ")
print(board[0])
print(board[1])
print(board[2])

#X fifth move
col = int(input("X player, select a column 1-3: "))
row = int(input("X player, select a row 1-3: "))
col -= 1
row -= 1

if board[row][col] == '-':
  board[row][col] = "X"
else:
  print("Oops, that spot was already taken. ")
print(board[0])
print(board[1])
print(board[2])
```

TRY IT

Directions: Now that you have the Tic-Tac-Toe program, go ahead and run it a few times to get reacquainted with it.

Now that you've had the chance to rerun the program, do you see anything that would enhance its functionality?

In the current state of coding, we can quickly determine that we have some repetitiveness in various places. Notice that we're constantly prompting each user for their selected position (O's third move, X's fifth move, etc.). We could loop those inputs for both the X and O players as part of the body of the loop. We will keep X player's input at the start, where we don’t have to determine if the spot has been taken. That essentially is letting the X user go first before the looping begins.

TRY IT

Directions: We have added a new element loop to the program code and used comments to call it out. Please add the looping for X and O player’s section for repeat player position input below. Then run the program to see the functionality.

```
board = [ ["-", "-", "-"],
          ["-", "-", "-"],
          ["-", "-", "-"] ]

print(board[0])
print(board[1])
print(board[2])

#X first move
col = int(input("X player, select a column 1-3: "))
row = int(input("X player, select a row 1-3: "))
col -= 1
row -= 1

board[row][col] = "X"
print(board[0])
print(board[1])
print(board[2])

for counter in range(1,5): #NEW CODE: loops for X and O players
    #O move
    col = int(input("O player, select a column 1-3: "))
    row = int(input("O player, select a row 1-3: "))
    col -= 1
    row -= 1

    if board[row][col] == '-':
      board[row][col] = "O"
    else:
      print("Oops, that spot was already taken. ")
    print(board[0])
    print(board[1])
    print(board[2])
    #X move
    col = int(input("X player, select a column 1-3: "))
    row = int(input("X player, select a row 1-3: "))
    col -= 1
    row -= 1

    if board[row][col] == '-':
      board[row][col] = "X"
    else:
      print("Oops, that spot was already taken. ")
    print(board[0])
    print(board[1])
    print(board[2])
```

Did you notice any change to the functionality of the program? Probably not, but that looping change has brought the code down from 130+ lines to 47 lines of code and has made it a much cleaner setup. However, if we really examine the code, it's not a big deal if the first check on the X player's input verifies if the spot is acceptable or not. We could wrap that into the loop as well.

The only difference between the X and O players is that the player alternates. The player that uses X takes turns 1, 3, 5, 7, and 9 while player O takes turns 2, 4, 6, and 8. We could base the player's turn on the number of the turn (even or odd) or we can simply alternate between the two players.

To do that, let's create a variable called `playerTurn` and allow this variable to change between the X and O player. By default, we'll set `playerTurn` to "X". We'll also set the `for` loop to loop 9 times and instead of the hardcoded value of X or O, we'll use the `playerTurn` variable instead.

TRY IT

Directions: Change the following lines of code using the directions and/or the "to" snippets.

Changes are explained from the top of the code to the bottom.

Change 1: Remove the following rows, because we will no longer need the "#X first move section" since we will have the `playerTurn` variable to switch between players.

Completely remove the following lines of code:

```
#X first move
col = int(input("X player, select a column 1-3: "))
row = int(input("X player, select a row 1-3: "))
col -= 1
row -= 1

board[row][col] = "X"
```

```
print(board[0])
print(board[1])
print(board[2])
```

Change 2: Add these rows where you just removed the #X first move section (immediately after the first time the board is printed out). This is initiating the variables `col` (column) and `row` and assigning them 0 before they are used in the loop. Also, we are creating and initializing `playerTurn` to “X”.

Add the following code:

```
col=0
row=0
playerTurn = "X"
```

Change 3: Change the loop initializing row to include 9 iterations instead of 5. Remember, if we want 9 interactions we need to include 10 in the `range()` function. This is due to us adding the code to automatically switch between players with a total of 9 iterations, as there are 9 possible moves in the game.

Change the following:

From:

```
for counter in range(1,5): #NEW CODE: loops for X and O players
```

To:

```
for counter in range(1,10):
```

Change 4: Delete the following row with a comment. Using the `playerTurn` variable we can use the rest of the original body of the loop within the original “O move” section for both players X and O now. Delete this line.

Delete the following line of code:

```
    #O move
```

Change 5: We will now include the `playerTurn` variable in the body of the loop. Change these following rows.

From:

```
    col = int(input("X player, select a column 1-3: "))
    row = int(input("X player, select a row 1-3: "))
```

To:

```
    col = int(input(playerTurn + " player, select a column 1-3: "))
    row = int(input(playerTurn + " player, select a row 1-3: "))
```

Change 6: Continue to use the `playerTurn` variable instead of X and O. Change the following row.

From:

```
board[row][col] = "X"
```

To:

```
board[row][col] = playerTurn
```

Change 7: Remove the “#X move” section of code from the program.

Delete the following lines of code.

```
#X move
    col = int(input("X player, select a column 1-3: "))
    row = int(input("X player, select a row 1-3: "))
    col -= 1
    row -= 1

    if board[row][col] == '-':
      board[row][col] = "X"
    else:
      print("Oops, that spot was already taken. ")
    print(board[0])
    print(board[1])
    print(board[2])
```

Change 8: Finally, we need to add the part that switches between players inside the `for` loop. Make sure this snippet of code is indented along with the rest of the body of the loop. Add this code to the bottom of the body of the loop.

Add the following code:

```
    if playerTurn =="X":
      playerTurn="O"
    else:
      playerTurn="X"
```

TRY IT

Directions: Make sure you have added, deleted, or updated the row(s) of the code above so your program will look like the code below. We added comments to the lines of code that were added or changed. Note that we have declared and initialized variables `col`, `row` and `playerTurn` since we will want to use it outside of the `for` loop. Once you have verified all additions/deletions/changes, go ahead and run the program and notice the updated functionality.

```
board = [ ["-", "-", "-"],
          ["-", "-", "-"],
          ["-", "-", "-"] ]

print(board[0])
print(board[1])
print(board[2])
col=0 #added on Change 2
row=0 #added on Change 2
playerTurn = "X" #added on Change 2

for counter in range(1,10): #updated on Change 3
    col = int(input(playerTurn + " player, select a column 1-3: ")) #updated on Change  5
    row = int(input(playerTurn + " player, select a row 1-3: ")) #updated on Change 5
    col -= 1
    row -= 1

    if board[row][col] == '-':
      board[row][col] = playerTurn #updated on Change 6
    else:
      print("Oops, that spot was already taken. ")
    print(board[0])
    print(board[1])
    print(board[2])

#following was added in Change 8
    if playerTurn =="X":
      playerTurn="O"
    else:
      playerTurn="X"
```

The gameplay is still working and it is easier and cleaner in the code using the overall variable `playerTurn` instead of switching back and forth from X to O individual code.

2. Adding Validation

Well, those changes in code have really helped simplify the code. But there is more we can do to make the game even better. The next thing we can do is have some error checking based on the row and column that was entered by the player. We want to check two separate things:

1. **Item 1**: We want to ensure that the value being entered by the player is between 1 and 3. Remember our two-dimensional list (2D) is 3 columns by 3 rows.
2. **Item 2**: We also want to ensure that if the position in the 2D list has an entry already there, we inform the player to choose again. Right now if a player selects a position that is already taken, they lose their turn.

Solution for Item #1

We can solve for item #1 above using a `while` loop to check for positions that are not valid.

EXAMPLE

```
    while (col < 1 or col > 3):
      col = int(input(playerTurn + " player, select a column 1-3: "))
      if (col < 1 or col > 3):
        print("The column must be between 1 and 3.")
```

In the code above, the `while` loop's condition states as long as the variable col's value is less than 1 or greater than 3, the statements in the body of the loop would be repeated. If the entry from the user for `col` was less than 1 or larger than 3, we'll output to the user that the column has to be between 1 and 3.

```
['-', '-', '-']
['-', '-', '-']
['-', '-', '-']
X player, select a column 1-3: 1
X player, select a row 1-3: 3
['-', '-', '-']
['-', '-', '-']
['X', '-', '-']
O player, select a column 1-3: 4
The column must be between 1 and 3.
O player, select a column 1-3: 1
O player, select a row 1-3: 1
['O', '-', '-']
['-', '-', '-']
['X', '-', '-']
X player, select a column 1-3:
```

Success! Now we can do the same for the row as well.

EXAMPLE

```
    while (row < 1 or row > 3):
      row = int(input(playerTurn + " player, select a row 1-3: "))
      if (row < 1 or row > 3):
        print("The row must be between 1 and 3.")
```

Here is the full code at this point. The solutions for item 1 (`while` loops from column and row) are commented below. Notice that we set the `col` and `row` to 0 before the `while` loop so with each iteration, the `col` and `row` will be reset to ensure that the user enters in an updated value each time.

EXAMPLE

```
board = [ ["-", "-", "-"],
         ["-", "-", "-"],
         ["-", "-", "-"] ]
print(board[0])
print(board[1])
print(board[2])
col=0
row=0
playerTurn = "X"

for counter in range(1,10):
   col=0
   row=0
   #here is the col while loops that addresses Item 1
   while (col < 1 or col > 3):
     col = int(input(playerTurn + " player, select a column 1-3: "))
     if (col < 1 or col > 3):
       print("The column must be between 1 and 3.")

   #here is the row while loops that addresses Item 1
   while (row < 1 or row > 3):
     row = int(input(playerTurn + " player, select a row 1-3: "))
     if (row < 1 or row > 3):
       print("The row must be between 1 and 3.")

   col -= 1
   row -= 1

   if board[row][col] == '-':
     board[row][col] = playerTurn
   else:
     print("Oops, that spot was already taken. ")
   print(board[0])
   print(board[1])
   print(board[2])
   if playerTurn =="X":
     playerTurn="O"
   else:
     playerTurn="X"
```

TRY IT

Directions: Go ahead and try running this program with the changes above and test for outside positions (1< or >3).

Solution for Item #2

Next, we can solve for the previous item #2 by adding a `while` loop that will keep looping the selection and use the conditional statement that we had to break out of the loop if the turn was valid. If not, we'll prompt the user to choose a different spot. We added a `while` loop that uses True as the conditional statement. We used True in this case as the set of statements should continue to prompt the user until a value was entered that was not taken up already. You'll notice that in the if statement that we have, if the move was a valid one (meaning there was a - in that spot), we would change the element to the `playerTurn` and then break out of the `while` loop. You'll also notice that if the user selected a spot that was already taken, the user retains their turn.

EXAMPLE

```
board = [ ["-", "-", "-"],
         ["-", "-", "-"],
         ["-", "-", "-"] ]
print(board[0])
print(board[1])
print(board[2])
col=0
row=0
playerTurn = "X"
for counter in range(1,10):
   while (True): #here is the while loop that addresses Item 2.
     col=0
     row=0
     while (col < 1 or col > 3):
       col = int(input(playerTurn + " player, select a column 1-3: "))
       if (col < 1 or col > 3):
         print("The column must be between 1 and 3.")
     while (row < 1 or row > 3):
       row = int(input(playerTurn + " player, select a row 1-3: "))
       if (row < 1 or row > 3):
         print("The row must be between 1 and 3.")
     col -= 1
     row -= 1
     if board[row][col] == '-':
       board[row][col] = playerTurn
       break; #here is the break statement that addresses Item 2
     else:
       print("Oops, that spot was already taken. Please select another spot.")
   print(board[0])
   print(board[1])
   print(board[2])
   if playerTurn =="X":
     playerTurn="O"
   else:
     playerTurn="X"
```

Here is an example output while we tested this updated code.

```
['-', '-', '-']
['-', '-', '-']
['-', '-', '-']
X player, select a column 1-3: 1
X player, select a row 1-3: 1
['X', '-', '-']
['-', '-', '-']
['-', '-', '-']
O player, select a column 1-3: 1
O player, select a row 1-3: 1
Opps, that spot was already taken. Please select another sopt.
O player, select a column 1-3:
```

TRY IT

Directions: Go ahead and try running this program again with the additional `while` loop and `break` statement and test some cases of positions that are already taken.

Checking Moves

Although this works, it's not an ideal approach. It would be better for us to use a variable as a sentinel value so that the loop can end correctly. A sentinel value is a value in which a loop should stop running. We'll create a variable called `validMove` and set it to False in the `for` loop. This way, for each iteration of the `for` loop, the `validMove` is set to False. Then, within the inner `while` loop, we'll change the expression to check if the `validMove` variable is False. If it is, the `while` loop will run. If the player's input is validated and the move is valid, we'll set `validMove` to True which will exit the `while` loop when the condition is checked again. Execution will then fall down to printing out the board, switching players and then back up to the `for` loop which will iterate and once again set `validMove` to False and start the process again.

EXAMPLE

```
board = [ ["-", "-", "-"],
          ["-", "-", "-"],
          ["-", "-", "-"] ]

print(board[0])
print(board[1])
print(board[2])
col=0
row=0
playerTurn = "X"

for counter in range(1,10):

    validMove = False #setting the validMove variable to False
    while (validMove == False): #while loop checking the validMove variable
      col=0
      row=0
      while (col < 1 or col > 3):
        col = int(input(playerTurn + " player, select a column 1-3: "))
        if (col < 1 or col > 3):
          print("The column must be between 1 and 3.")
      while (row < 1 or row > 3):
        row = int(input(playerTurn + " player, select a row 1-3: "))
        if (row < 1 or row > 3):
          print("The row must be between 1 and 3.")
      col -= 1
      row -= 1
      if board[row][col] == '-':
        board[row][col] = playerTurn
        validMove=True; #setting validMove to True to exit loop
      else:
        print("Oops, that spot was already taken. Please select another spot.")

    print(board[0])
    print(board[1])
    print(board[2])

    if playerTurn =="X":
      playerTurn="O"
    else:
      playerTurn="X"
```

TRY IT

Directions: Add the changes (`validMove` checks) above and try the program again. The new lines of code are commented.

There is still room for improvement, such as checking if the player has won or not. We'll get further into that in the next challenge with functions and methods.

To see the final version of this program visit **Sophia's Python code page**

SUMMARY

In this lesson, we improved the **Tic-Tac-Toe** program by using **loops** for player input. Using loops greatly reduced the large blocks of repetitive code from our first version of the game. We then created nested loops to check for input position **validation** within the game board and whether or not the chosen position was already taken. If either validation was invalid, an output message was sent to the player and they were asked to try again.

Best of luck in your learning!

Source: THIS CONTENT AND SUPPLEMENTAL MATERIAL HAS BEEN ADAPTED FROM "PYTHON FOR EVERYBODY" BY DR. CHARLES R. SEVERANCE ACCESS FOR FREE AT **www.py4e.com/html3/** LICENSE: **CREATIVE COMMONS ATTRIBUTION 3.0 UNPORTED**.