



# Girls' Programming Network

## ***Tic-Tac-Toe***

SplootCode Edition

## **Tutors Only**

*Create a 2 player Tic Tac Toe game to play with  
your friends!*

# Part 0: Setting up

## Task 0.1: Making a SplootCode project

- Make sure the student opens **Chrome** (Firefox is slower for SplootCode)
- Open <https://app.splootcode.io> and log in.

## Task 0.2: You've got a blank space, so write your name!

Encourage the student to write her name at the top of the file, in a comment.  
Type # for a comment.

## ☑ CHECKPOINT ☑

**If you can tick all of these off you can go to Part 1:**

- ☐ You should have a file called `tic_tac_toe.py`
- ☐ Your file has your name at the top in a comment
- ☐ Run your file with F5 key and it does nothing!!

## ★ BONUS 0.3: Customised Welcome ★

- If the student's program can't find the *name* variable, check that they've spelled the variable in the same way, and that the capitalization is the same.

# Part 1: Welcome to Tic-Tac-Toe!

## Task 1.1: Storing the board

In SplootCode, lists and function arguments stack vertically instead of using commas to separate them.

The student's code should look like this.

```
assign v board = list (
  ' ' ' '
  ' ' ' '
  ' ' ' '
  ' ' ' '
  ' ' ' '
  ' ' ' '
  ' ' ' '
  ' ' ' '
)
```

### Hint

Make sure the student has called the list **board**, as opposed to **my\_list**.

## Task 1.2: Printing the board

At this stage, the code should look like this:

```
f print ( '-' x 13 )
f print ( ' | ' + v board item ( 0 ) + ' | ' + v board item ( 1 ) + ' | ' + v board item ( 2 ) + ' | ' )
f print ( '-' x 13 )
f print ( ' | ' + v board item ( 3 ) + ' | ' + v board item ( 4 ) + ' | ' + v board item ( 5 ) + ' | ' )
f print ( '-' x 13 )
f print ( ' | ' + v board item ( 6 ) + ' | ' + v board item ( 7 ) + ' | ' + v board item ( 8 ) + ' | ' )
f print ( '-' x 13 )
```

### Hint: Concatenation

It's also possible to put multiple things in a print statement as separate parameters, and that they will have spaces inserted between them, like this.

```
f print ( ' | '
      { v board item ( 0 )
        ' | '
        v board item ( 1 )
        ' | '
        v board item ( 2 )
        ' | '
      }
```

### Task 1.3: Print Test!

After testing their board with different symbols in it, make sure students reset it to just having spaces.

### ✓ CHECKPOINT ✓

**If you can tick all of these off you can go to Part 2:**

- ☐ Use a list to create the board
- ☐ Print your empty playing board

## Part 2: Enter The First Move

### Task 2.1: What symbol are you?

The code should look like this:

```
assign v symbol = '0'
```

### Task 2.2: Which spot do you want to choose?

The student's code should be as above, plus this line:

```
assign v square = f input ( 'Which square do you want your symbol to go in? ' )
```

### Hint

Leave a space after the question mark, so that the prompt isn't immediately followed by the user's input (as seen here):


Needs a space!

which square do you want your symbol to go in? oh no there's no space!

### Task 2.3: Find the square on the board

The code should be the same as above, but the square question becomes:

```
assign v symbol = '0' → "0" (str)

assign v square = f input( 'Which square do you want your symbol to go in? ' )
assign v square_index = f int( v square ) → 3 (int)
```

#### Hint

Some students may combine the two lines into one, which is fine:

```
assign v square_index = f int( f input( 'Which square do you want your symbol to go in? ' ) )
```

However, if they name the variable **square** (instead of **square\_index**), they could get confused when later instructions refer to **square\_index**.

### Task 2.4: Update list with player's symbol

The student's full code (minus printing the board) should now be:

```
assign v symbol = '0' → "0" (str)
assign v square_index = f int( f input( 'Which square do you want your symbol to go in? ' ) )
assign v board item( v square_index ) = v symbol → "0" (str)
```

## ✓ CHECKPOINT ✓

If you can tick all of these off you can go to Part 3:

- ☐ The player's chosen symbol is stored in a variable
- ☐ The spot the player wants to move is stored in a variable
- ☐ Update the list with player's symbol

## ★ BONUS 2.5: Welcome the players ★

If the student does this bonus, the code should look something like this:

```
assign v player_O = f input ( ' Who is playing naughts? ' )  
assign v player_X = f input ( ' Who is playing crosses? ' )  
f print ( ' Welcome ' + v player_O + ' , your symbol is O! ' )  
f print ( ' Welcome ' + v player_X + ' , your symbol is X! ' )
```



## Part 3: Creating a print function

We updated the `board` list, but to actually show our updated board, we need to `print` the board again. We already have code that `prints`, so to avoid repeating that code, let's create a function!

### Task 3.1: Define your function!

The top of the student's code should now look like this:

```
# Firstname Lastname

def print_board(board):

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

### Task 3.2: Fill in your function body

The code should look like this:

```
function v print_board ( v board )
    f print ( '-' x 13 )
    f print ( '|' + v board item ( 0 ) + '|' + v board item ( 0 ) + '|' + v board item ( 0 ) + '|' )
    f print ( '-' x 13 )
    f print ( '|' + v board item ( 3 ) + '|' + v board item ( 4 ) + '|' + v board item ( 5 ) + '|' )
    f print ( '-' x 13 )
    f print ( '|' + v board item ( 6 ) + '|' + v board item ( 7 ) + '|' + v board item ( 8 ) + '|' )
    f print ( '-' x 13 )
```

### Hint: What is the function body?

- Make sure that the line assigning the `board` variable **is not inside the function**. The board should be initialised separately, outside of the function.
- Students may have trouble moving their code into the function. Select the code with shift + arrow keys or shift + click. Then use ctrl-c and ctrl-v to copy paste the code.

### Task 3.3: Let's call our function!

After this step, the code should now look like this:

```
Called 1 times
function v print_board ( v board )
    f print ( '-' x 13 ) → None
    f print ( '|' + v board item ( 0 ) + '|' + v board item ( 0 ) + '|' + v board it
    f print ( '-' x 13 ) → None
    f print ( '|' + v board item ( 3 ) + '|' + v board item ( 4 ) + '|' + v board it
    f print ( '-' x 13 ) → None
    f print ( '|' + v board item ( 6 ) + '|' + v board item ( 7 ) + '|' + v board it
    f print ( '-' x 13 ) → None

assign v board = list ( ' ' ) → (list)
    ' '
    ' '
    ' '
    ' '
    ' '
    ' '
    ' '

f print_board ( v board ) → None

assign v symbol = 'O' → "O" (str)
assign v square_index = f int ( f input ( 'Which square do you want your symbol to go in? ' ) )
assign v board item ( v square_index ) = v symbol → "O" (str)
```

### Task 3.4: Reveal the updated board!

The code should now be as above, plus a final line that calls the function again:

```
f print_board ( v board )
```

## ✓ CHECKPOINT ✓

If you can tick all of these off you can go to Part 4:

- ☐ Define a function called `print_board`
- ☐ Print your empty playing board at the beginning of your game
- ☐ Print your updated playing board after your first move





## Part 4: Taking Turns

### Task 4.1: Your turn!

The student should add a print statement after they set the symbol:

```
f print('The current player is ' + v symbol + '!') → None
assign v square_index = f int(f input('Which square do you want your symbol to go in? '))
assign v board item(v square_index) = v symbol → "O" (str)

f print_board(v board) → None
```

### Task 4.2: You get a turn! And you get a turn! And you get a turn!

The student's code should be as above, plus this `if` statement at the end:

```
if v symbol == 'O' → True
    assign v symbol = 'X' → "X" (str)
```

### Task 4.3: Switch back!

The `if` statement now gets a second part to switch back to naughts:

```
if v symbol == 'O' → True
    assign v symbol = 'X' → "X" (str)
else
    assign v symbol = 'O'
```

**WARNING!** If the student writes `if` twice, both conditions will be true! The symbol will be changed to crosses, then immediately back to naughts:

```
if v symbol == 'O' → True
    assign v symbol = 'X' → "X" (str)

if v symbol == 'X' → True
    assign v symbol = 'O' → "O" (str)
```

Wrong!  
Don't do this!

So be sure to use either `else` or `elif`, so that the symbol is only changed once.

## ✓ CHECKPOINT ✓

If you can tick all of these off you can go to Part 5:

- ☐ Start playing as Noughts
- ☐ Tell the players whose turn it is
- ☐ Switch players at the bottom of your code

## Part 5: Wait a while to win

### Task 5.1: Game Over?

After initializing the board, the student's code should initialize a new variable:

```
assign v game_over = False → False
```

### Task 5.2: Did I win yet?

After adding the while loop, the code should look like this.

```
Repeated 7 times ● ———— < 1 >
while not v game_over → True
    f print('The current player is ' + v symbol + '!!') → None
    assign v square_index = f int(f input('Which square do you want
    assign v board item(v square_index) = v symbol → "O" (str)

    f print_board(v board) → None

    if v symbol == 'O' → True
        assign v symbol = 'X' → "X" (str)
    else
        assign v symbol = 'O'
```

### Hint

Make sure the while loop condition is **not** game\_over - don't miss the **not** !

## ✓ CHECKPOINT ✓

**If you can tick all of these off you can go to Part 6:**

- ☐ You have set your `game_over` variable
- ☐ You have built your while loop
- ☐ You have run your code to make sure the indents are correct and get no errors

For Loops

## Part 6: Winner winner, tic tac dinner!

### Task 6.1: Where are the winners

The answers for this section are:

Rows	0, 1, 2	3, 4, 5	6, 7, 8
Columns	0, 3, 6	1, 4, 7	2, 5, 8
Diagonals	0, 4, 8	2, 4, 6	--

### Task 6.2: Functions again!

The top of the student's code should get a new function definition:

Called 0 times

```
function v check_winner ( v board )
```

### Task 6.3: One function, two options!

The student can fill in `check_winner(board)` in two ways, (1) Blue or (2) Orange. Depending on their choice, their function should look something like this:

#### Option 1: If statements

```
function v check_winner ( v board )
  if ( v board item ( 0 ) == v board item ( 1 ) == v board item ( 2 ) ≠ ' ' )
    return True
  else if ( v board item ( 3 ) == v board item ( 4 ) == v board item ( 5 ) ≠ ' ' )
    return True
  else if ( v board item ( 6 ) == v board item ( 7 ) == v board item ( 8 ) ≠ ' ' )
    return True
  else if ( v board item ( 0 ) == v board item ( 3 ) == v board item ( 6 ) ≠ ' ' )
    return True
  else if ( v board item ( 1 ) == v board item ( 4 ) == v board item ( 7 ) ≠ ' ' )
    return True
  else if ( v board item ( 2 ) == v board item ( 5 ) == v board item ( 8 ) ≠ ' ' )
    return True
  else if ( v board item ( 0 ) == v board item ( 4 ) == v board item ( 8 ) ≠ ' ' )
    return True
  else if ( v board item ( 2 ) == v board item ( 4 ) == v board item ( 6 ) ≠ ' ' )
    return True
  return False
```

Note that the triples are exactly the same as the winning triples identified in the last exercise. **Don't forget to check that the triples aren't spaces!**

#### Option 2: For loop and lists

```

assign v winning_combos = list (
  list (0)
  list (1)
  list (2)
  list (3)
  list (4)
  list (5)
  list (6)
  list (7)
  list (8)
  list (0)
  list (3)
  list (6)
  list (1)
  list (4)
  list (7)
  list (2)
  list (5)
  list (8)
  list (0)
  list (4)
  list (8)
  list (2)
  list (4)
  list (6)
)

```

Again, notice how the triples are the same as those identified in the last exercise.

```

function v check_winner ( v board )

    for v combo in v winning_combos
        assign v symbol0 = v board item ( v combo item ( 0 ) )
        assign v symbol1 = v board item ( v combo item ( 1 ) )
        assign v symbol2 = v board item ( v combo item ( 2 ) )
        if v symbol0 == v symbol1 == v symbol2 ≠ ' '
            return True

    return False

```

### ✓ CHECKPOINT ✓

**OPTION 1: If you can tick all of these off you can go to Part 7:**

- ☐ You have created the `check_winner` function
- ☐ You return the result from the `check_winner` function
- ☐ Your `check_winner` function checks every possible way to win

### ✓ CHECKPOINT ✓

**OPTION 2: If you can tick all of these off you can go to Part 7:**

- ☐ You have created `winning_combos` with eight tuples in it
- ☐ You have created the `check_winner` function
- ☐ You return the result from the `check_winner` function
- ☐ Your `check_winner` function checks every possible way to win

## Part 7: Declare the winner

### Task 7.1: Check whether the game has been won

The main game loop should now look like this:

```
while not v game_over
    f print ( 'The current player is ' + v symbol + '!' )
    assign v square_index = f int ( f input ( 'Which square do you want your symbol to go in? ' ) )
    assign v board item ( v square_index ) = v symbol

    f print_board ( v board )
    assign v game_over = f check_winner ( v board )

    if v symbol == '0'
        assign v symbol = 'X'
    else
        assign v symbol = '0'
```

### Task 7.2: Declare who won

The final code for the base game is as follows (not including functions as above)

```
assign v board = list (
    ( ' ' ' ' )
    ( ' ' ' ' )
    ( ' ' ' ' )
    ( ' ' ' ' )
    ( ' ' ' ' )
    ( ' ' ' ' )
    ( ' ' ' ' )
    ( ' ' ' ' )
)

assign v game_over = False
f print_board ( v board )

assign v symbol = '0'
```



```

while not v game_over
    f print( 'The current player is ' + v symbol + '!' )
    assign v square_index = f int( f input( 'Which square do you want your symbol to go in? ' ) )
    assign v board item( v square_index ) = v symbol

    f print_board( v board )
    assign v game_over = f check_winner( v board )
    if v game_over
        f print( v symbol + ' won! Congratulations!' )

    if v symbol == '0'
        assign v symbol = 'X'
    else
        assign v symbol = '0'

```



## Girls' Programming Network

***Tic-Tac-Toe***

***Extensions***

**TUTORS ONLY**

*Make your 2 player Tic Tac Toe game even better!*

## Extension 8: You can't go there!

At the moment the game lets you cheat by playing in a spot someone else has already taken! And the game breaks if you enter a number bigger than 8!

Let's fix it so you can only play in spots that actually exist! And not ones that are taken.

### Task 8.1: What spaces are free?

The answers for this exercise are:

X	O	X

free\_squares =  
[3,4,5,6,7,8]

O		
	X	
		O

free\_squares =  
[1,2,3,5,6,7]

X	X	
O	O	

free\_squares =  
[2,5,6,7,8]

### Task 8.2: What spaces are free?

The student should initialize the list just before the main loop:

```
assign v free_squares = list ( 0 )  
                                ( 1 )  
                                ( 2 )  
                                ( 3 )  
                                ( 4 )  
                                ( 5 )  
                                ( 6 | )  
                                ( 7 )  
                                ( 8 )
```

```
while not v game_over  
    f print ( ' The current player is ' + v symbol + ' ! ' )  
    assign v square_index = f int ( f input ( ' Which square do
```

### Task 8.3: Keeping track of spaces

Inside the main game loop, the student needs to add two lines after they print the board:

```

while not v game_over
    f print ( 'The current player is ' + v symbol + '!' )
    assign v square_index = f int ( f input ( 'Which square do you want your symbol to go in?' ) )
    assign v board item ( v square_index ) = v symbol

    f print_board ( v board )
    v free_squares . remove ( v square_index ) ←
    f print ( v free_squares ) ←

    assign v game_over = f check_winner ( v board )
    if v game_over

```

### Task 8.4: Is that even allowed?

At the end of this task, the student should have added these two lines:

```

f print ( 'The current player is ' + v symbol + '!' )
assign v square_index = f int ( f input ( 'Which square do you want your symbol to go in?' ) )
if v square_index not in v free_squares
    f print ( 'Hey, that square isn't allowed!' )
    assign v board item ( v square_index ) = v symbol

```

HINT: These two versions of the if do exactly the same thing:

```

if v square_index not in v free_squares
    f print ( 'Hey, that square isn't allowed!' )

if not v square_index in v free_squares
    f print ( 'Hey, that square isn't allowed!' )

```

## Task 8.5: Try that again

The final code for Extension 8 has the main game loop as follows, with the new parts highlighted:

```
while not v game_over
    f print('The current player is ' + v symbol + '!' )
    assign v square_index = f int( f input('Which square do you want your symbol to go in? ' ) )

    if v square_index not in v free_squares
        f print('Hey, that square isn't allowed!')
        continue

    assign v board item( v square_index ) = v symbol

    f print_board( v board )
    v free_squares .remove( v square_index )
    f print( v free_squares )

    assign v game_over = f check_winner( v board )
    if v game_over
        f print( v symbol + ' won! Congratulations!' )

    if v symbol == '0'
        assign v symbol = 'X'
    else
        assign v symbol = '0'
```

## ✓ CHECKPOINT ✓

If you can tick all of these off you've finished Extension 8:

- ☐ Your game doesn't let you play in a square someone already filled
- ☐ Your game tells the player if they chose a bad square and starts their turn again

☐ Your game doesn't break when you choose squares like 99 or -5

## Extension 9: It's a tie!

When we hit 9 moves, the board is filled and there's no winner. We have a tie!

### Task 9.1: Count the turns

Tutor note: The example code given assumes the student is *only* doing Extension 9, without doing Extension 8 first.

The student needs to add two lines for this task:

```
assign v counter = 0
```

```
while not v game_over
```

```
    f print ( 'The current player is ' + v symbol + '!' )
```

```
    assign v square_index = f int ( f input ( 'Which square do you want
```

```
    if v square_index not in v free_squares
```

```
        f print ( 'Hey, that square isn't allowed!' )
```

```
        continue
```

```
    assign v board item ( v square_index ) = v symbol
```

```
    f print_board ( v board )
```

```
    assign v counter = v counter + 1
```

```
    v free_squares . remove ( v square_index )
```

```
    f print ( v free_squares )|
```



## Task 9.2: Check for a tie

After this task, the code should look like this (new lines highlighted):

```
assign v game_over = f check_winner ( v board )
if v game_over
  f print ( v symbol + 'won! Congratulations!' )
else if v counter == 9
  f print ( 'It's a tie!' )
  break
```

### ✓ CHECKPOINT ✓

**If you can tick all of these off you've finished Extension 9:**

- ☐ The game ends if the board is all filled up
- ☐ The game prints that it's a tie if no one has won at the end of the game

## Extension 10: Coin Toss

At the moment the same symbol always starts first. Let's make it randomly chooses who goes first!

### Task 10.1: This is random!

Tutor note: The code snippets are assuming that the student is *only* doing Extension 10, and hasn't done Extension 8 or 9.

This task is super easy, just add the import at the top of the file:

```
import random
```

### Task 10.2: Who will go first?

The student can solve this task with `random.choice()`:

```
assign v symbol = ( v random . choice ( list ( 'X' ) )  
                  ( 'O' ) )
```

### Task 10.3: Tell us who's next!

Depending on their existing code, **the student may not need to change their program at all** in order to announce the next player:

```
assign v symbol = ( v random . choice ( list ( 'X' ) )  
                  ( 'O' ) )  
  
while not v game_over  
    f print ( 'The current player is ' + v symbol + '!' )  
    assign v square_index = f int ( f input ( 'Which square do you want your symbol  
  
    if v square_index not in v free_squares  
        f print ( 'Hey, that square isn't allowed!' )  
        continue
```

They probably already have this line!

Otherwise, a `print` statement will do the job.

## ✓ CHECKPOINT ✓

**If you can tick all of these off you've finished Extension 10:**

☐ Your game randomly chooses which symbol will start

☐ Your game announces the winner of the coin toss

# Extension 11: A game that knows your name!

It would be better if the game actually referred to you by name, not just your symbol!

## Task 11.1: Prepare yourself!

Tutor note: These code snippets assume that the student already completed Extension 10.

Also, make sure the student has done **Bonus 2.5** where the program asks for the players names.

## Task 11.2: Who's there

**After** the function definitions, but **before** the main game loop, the student's code should now look something like this:

```
assign v game_over = False
f print_board ( v board )|
assign v counter = 0

assign v player_O = f input ( 'Who is playing naughts? ' )
assign v player_X = f input ( 'Who is playing crosses? ' )
f print ( 'Welcome ' + v player_O + ', your symbol is O!' )
f print ( 'Welcome ' + v player_X + ', your symbol is X!' )

assign v symbol = v random . choice ( list ( 'X' )
                                     ( 'O' ) )

while not v game_over
    f print ( 'The current player is ' + v symbol + '!' )
    assign v square_index = f int ( f input ( 'Which square do you want your sy
```

## Task 11.3: It's your turn!

After this task, the code becomes:

```
assign v symbol = v random.choice( list( 'X' )
                                     'O' )
if v symbol == 'O'
    assign v current_player = v player_O
else
    assign v current_player = v player_X

while not v game_over
    f print( 'The current player is ' + v current_player + ' who is playing ' + v symbol )
    assign v square_index = f int( f input( 'Which square do you want your symbol to go in? ' ) )
```

### Task 11.4: Who's next?

Inside the main game loop, the student can take advantage of the existing if statements, changing the **current\_player** at the same time as the **symbol**:

```
if v symbol == 'O'
    assign v symbol = 'X'
    assign v current_player = v player_X
else
    assign v symbol = 'O'
    assign v current_player = v player_O
```

### Task 11.5: Who's won?

This one is simple, the student just needs to change the **symbol** variable to **current\_player**:

```
assign v game_over = f check_winner( v board )
if v game_over
    f print( v current_player + ' won! Congratulations!' )
else if v counter == 9
    f print( 'It's a tie!' )
    break
```

## ✓ CHECKPOINT ✓

If you can tick all of these off you've finished Extension 11:

- ☐ The game prints out the name of the player who owns the symbol each turn
- ☐ The game keeps track of which players turn it is.

# Extension 12: Random computer player

Right now we need a friend to play, but what if we want to play when no one else is around? Let's make very basic computer player. It will randomly choose a place to put its symbols!

In this game if one of the names entered is computer, then we will choose a random square for the computer to fill. (I hope none of your friends names are computer!)

## Task 12.1: Prepare yourself!

Make sure the student has done the other extensions:

1. **Extension 8: You Can't Go There**
2. **Extension 10: Coin Toss**
3. **Extension 11: A Game that Knows Your Name!**

The code snippets below assume that the student has completed those extensions.

## Task 12.2: My name is computer

The student needs to add an `if` statement which checks if it's the computer's turn. If so, the computer chooses a free square randomly:

```
if v.current_player == 'computer':  
    assign v.square_index = v.random.choice(v.free_squares)
```

## Task 12.3: I'm no robot!

Make sure that the student **only** indents the code that asks the human player for their square!

The code that places the symbol on the board should still be out of the `if/else` block:

```
while not v.game_over:  
    f print('The current player is ' + v.current_player + ' who is playing ' + v.symbol )  
    if v.current_player == 'computer':  
        assign v.square_index = v.random.choice(v.free_squares)  
    else:  
        assign v.square_index = f int(f input('Which square do you want your symbol to go in? '))  
  
    if v.square_index not in v.free_squares:  
        f print('Hey, that square isn't allowed!')  
        continue
```

## ✓ CHECKPOINT ✓

**If you can tick all of these off you've finished Extension 12:**

- ☐ If you say a computer is playing the game randomly chooses moves for the computers turn.
- ☐ You print out the free squares each turn and it gets smaller as the game goes on.
- ☐ The human player still gets to choose a move on their turn.

## ★ BONUS 12.4: Smarter Computer ★

BONUS 12.4 can be approached in multiple ways. The student will try to code:

- A computer player that will play in an empty spot that would complete their line
- Will block the opponent from winning if the opponent already has two in a row
- All the other times it can still play randomly

General debugging principles are useful here; if the student is stuck, encourage her to add print statements for important variables, to check that everything is working as she expects.