



# Lab 02: Find Four Game

## **Overview**

This lab is designed to introduce students to 2D arrays by recreating the game **Find Four** (brand name "Connect Four"). This will require students to loop through and manipulate sequential data structures.

# **Specification**

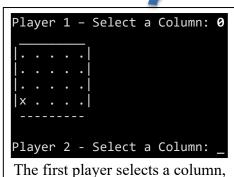
The program should be driven by the **findfour** module and its functions.

When started, the program should display a welcome message, then ask the user for a width and height of the playing board. Once the user has entered the dimensions, the player's chips should be displayed, followed by the empty board, and the first player should be prompted for a move.

Player 1: x player 2: 0

### **Gameplay**

Players take turns adding chips to the board, which drop to the lowest open row. Once one player achieves four vertical or horizontal chips in a row (not diagonal), that player wins. Play proceeds as follows.



and the new board is displayed.

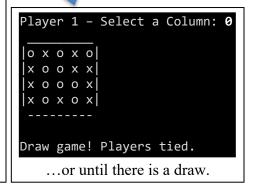
The next player is prompted.

```
Player 1 - Select a Column: 0

| x . . . . |
| x . . . . |
| x . o . . |
| x o x o o |
------

Player 1 won the game!
```

Play continues to alternate until one player wins...



## **Data Storage**

Elements in the container should be accessible via row-major indexing (board[row][column]). In addition, the data should be stored so that row zero is the bottom of the board, i.e.:

Make sure you test as you complete this assignment; otherwise, the unit tests used for evaluation will fail!

### **Error Handling**

This assignment requires students to deal with user error and handle malformed input. The number of columns and rows must be in the range [4, 25]. Note: error handlings should be handled during input, and not in the method proscribed to update the board (see *Functions* section). The program must handle error cases as follows.

#### **Invalid Dimensions**

```
Welcome to Find Four!
                                      Welcome to Find Four!
Enter height of board (rows): 1
                                      Enter height of board (rows): 5
Error: height must be at least 4!
                                      Enter width of board (columns): 9001
Enter height of board (rows): -1
                                       Error: width can be at most 25!
Error: height must be at least 4!
                                      Enter width of board (columns): 3
Enter height of board (rows): 42
                                      Error: width must be at least 4!
                                      Enter width of board (columns): wUT
Error: height can be at most 25!
Enter height of board (rows):
                                       Error: not a number!
                                      Enter width of board (columns):
```

#### Invalid Column Entry

```
Player 2 - Select a Column: 0
Player 2 - Select a Column: 0
                                                                      Player 2 - Select a Column: 0
x x o . .
                                      x o . .
                                                                       x x o . .
x \circ x \circ o
                                     o \times o o
                                                                       x \circ x \circ ol
Player 1 - Select a Column: 0
                                   Player 1 - Select a Column: 5
                                                                      Player 1 - Select a Column: fOo
                                   rror: no such column!
                                                                      Error: not a number!
Error: column is full!
                                                                      Player 1 - Select a Column:
Player 1 - Select a Column:
                                   Player 1 - Select a Column:
```

### **Required Functions**

```
get initial board(rows: int, columns: int) -> list[list[str]]
Returns a list of size rows, where each entry is itself a list of size columns, where each entry contains the value '.'.
print_board(board: list[list[str]])
Prints a copy of the board, where board is a list of list objects, each entry a one-character str object.
```

```
insert_chip(board: list[list[str]], column: int, chip: str) -> int
Places a chip in the column of the board of the chip type. This method should find the next available spot in that column,
if any. This method returns the row in which the chip settles.
```

```
is_win_state(chip: str, board: list[list[str]], row: int, column: int) -> bool
This method checks if the player represented by specified chip type has won the game by looking on the board at the
position (row, column). If this is a win for the player, returns True; otherwise, returns False.
```

```
is board full(board: list[list[str]]) -> bool
This method checks if the board is full. If it is full, returns True; otherwise, returns False.
```

## **Submission**

NOTE: Output must match example output exactly. Otherwise, your submission will not receive full credit!

Files: FindFour.py Method: Submit on ZyLabs

# **Sample Output**

```
Welcome to Find Four!
_____
Enter height of board (rows): 2
Error: height must be at least 4!
Enter height of board (rows): 4
Enter width of board (columns): 5
|. . . . .
|. . . . .|
|\cdot \cdot \cdot \cdot|
|. . . . . |
Player 1: x
Player 2: o
Player 1 - Select a Column: `0
Error: not a number!
Player 1 - Select a Column: 0
|....|
|. . . . . |
|x . . . . |
Player 2 - Select a Column: 5
Error: no such column!
Player 2 - Select a Column: 3
. . . . .
|\cdot|\cdot|\cdot|
|. . . . .|
|x . . o .|
Player 1 - Select a Column: 0
|. . . . .|
|x . . . .
x . . o .
Player 2 - Select a Column: -1
Error: no such column!
Player 2 - Select a Column: 1
|. . . . . |
|x . . . . |
|x o . o .|
```

```
|x . . . . |
|x . . . . |
|x o . o .|
Player 2 - Select a Column: 4
|x . . . . |
|x . . . . |
|x \circ . \circ \circ|
Player 1 - Select a Column: 2
| . . . . . |
|x . . . . |
|x . . . . |
|x \circ x \circ o|
Player 2 - Select a Column: 2
|x . . . .
|x . o . .|
|x \circ x \circ o|
Player 1 - Select a Column: 0
|x . . . .
|x . . . . |
|x . o . .|
|x \circ x \circ o|
Player 1 won the game!
Process finished with exit code 0
```

Player 1 - Select a Column: 0