```
1 from tictactoe_board import *
 2
3 def main():
       the_board = Tictactoe_board(['XOX',
 4
                                     'OXO',
 5
                                     'X00'])
 6
7
       print(the_board)
       print("The winner is %s" % the_board.get_winner())
 8
       print()
 9
10
       the_board.place_piece(2, 0, '0')
11
       print(the_board)
12
       print("The winner is %s" % the_board.get_winner())
13
14
15 if __name__ = "__main__":
       main()
16
17
```

```
File - /Users/chrishegangkim/Desktop/Union College/Spring 2023/CSC 120/Kim_Lab5/testing.py
 1 """
 2 Testing utilities. Do not modify this file!
 3 """
 4
 5 VERBOSE = True
 6 num_pass = 0
 7 num_fail = 0
 9 def assert_equals(msq, expected, actual):
10
11
        Check whether code being tested produces
12
        the correct result for a specific test
13
        case. Prints a message indicating whether
14
       it does.
15
        :param: msg is a message to print at the beginning.
16
        :param: expected is the correct result
17
        :param: actual is the result of the
18
        code under test.
        11 11 11
19
20
        if VERBOSE:
21
            print(msg)
22
23
        global num_pass, num_fail
24
25
        if expected = actual:
26
            if VERBOSE:
27
                print("PASS")
28
            num_pass += 1
29
        else:
30
            if not VERBOSE:
31
                print(msq)
32
            print("**** FAIL")
33
            print("expected: " + str(expected))
34
            print("actual: " + str(actual))
35
            if not VERBOSE:
36
                print("")
37
            num_fail += 1
38
39
        if VERBOSE:
            print("")
40
41
```

```
42
43 def fail on error(msq.err):
44
45
       if run-time error occurs, call this to insta-fail
46
47
       :param msq: message saying what is being tested
48
       :param err: type of run-time error that occurred
49
50
       qlobal num_fail
51
       print(msq)
52
       print("**** FAIL")
53
       print(err)
       print("")
54
       num_fail += 1
55
56
57
58 def start_tests(header):
59
       Initializes summary statistics so we are ready to run
60
   tests using
       assert_equals.
61
62
       :param header: A header to print at the beginning
63
       of the tests.
64
65
       global num_pass, num_fail
66
       print(header)
67
       for i in range(0,len(header)):
           print("=",end="")
68
       print("")
69
70
       num_pass = 0
71
       num_fail = 0
72
73 def finish_tests():
74
75
       Prints summary statistics after the tests are complete.
76
       print("Passed %d/%d" % (num_pass, num_pass+num_fail))
77
       print("Failed %d/%d" % (num_fail, num_pass+num_fail))
78
79
       print()
80
```

```
1 Lab Question 1
2 __row_as_string
3 __three_in_row
4 __is_winner
5
6 Lab Question 2
7 self.__board
8
9 Lab Question 3
10 With the given rows, lists of '0' and 'X' are appended to the board, but it appears as X | 0 | X with new lines
11 in the Tictactoe_board object.
```

```
File - /Users/chrishegangkim/Desktop/Union College/Spring 2023/CSC 120/Kim_Lab5/tictactoe_board.py
 1 """
 2 defines the behavior of a tic-tac-toe board
 3 """
 4
 5 \text{ NUM}_{ROWS} = 3
 6
 7 class Tictactoe_board:
 8
 9
        def __init__(self, rows):
10
11
            Constructor. Creates a tictactoe board with given cell
    values.
12
            If no initial cell values are given, creates an empty
   tictactoe board.
13
14
            :param rows: A list of three 3-character strings,
   where each character
            is either 'X', 'O', or ' '. Each of the
15
16
            3-character strings represents a row of the tictactoe
   board.
            Example: [" X ", "0 0", "XX0"] is the board
17
               | X |
18
19
             0 | | 0
20
21
22
            X \mid X \mid 0
            11 11 11
23
24
            self.__board = []
            if rows is None:
25
                 empty_row = [' ', ' ', ' ']
26
27
                 for i in range(NUM_ROWS):
28
                     self.__board.append(empty_row)
29
            else:
30
                 for i in range(NUM_ROWS):
31
                     row = []
32
                     for j in range(NUM_ROWS):
33
                          row.append(rows[i][j])
34
                     self.__board.append(row)
35
        def place_piece(self, i, j, piece):
36
37
```

```
File - /Users/chrishegangkim/Desktop/Union College/Spring 2023/CSC 120/Kim Lab5/tictactoe board.py
            Places a piece (either 'X' or '0') on the board.
38
39
40
            :param i: The row in which to place a piece (0, 1, or
   2)
            :param j: The column in which to place a piece (0, 1,
41
   or 2)
42
            :param piece: The piece to place ('X' or '0')
43
44
            self.__board[i][j] = piece
45
46
        def clear_cell(self, i, j):
47
48
            Clears a cell on the tictactoe board.
49
50
            :param i: The row of the cell to clear
51
            :param j: The column of the cell to clear
52
53
            self.place_piece(i, j, ' ')
54
55
        def __row_as_string(self,row):
56
57
            returns row in a format suitable for printing
58
            :param row: row of board as list of strings
59
            :return: row in prettified string format
60
            str = ''
61
62
            for column in row[:len(row)-1]:
63
                str += column + ' | '
64
            str += row[len(row)-1]
65
            return str
66
        def __str__(self):
67
68
69
            Produces a string representation of a board, returns
   it.
70
71
            :return: The string version of the board.
            11 11 11
72
73
            result = ''
74
            for row in self.__board[:len(self.__board)-1]:
                result += self.__row_as_string(row)
75
```

```
File - /Users/chrishegangkim/Desktop/Union College/Spring 2023/CSC 120/Kim Lab5/tictactoe board.py
                  result += '\n----\n'
 76
 77
             result += self.__row_as_string(self.__board[len(self.
     __board)-1])
 78
             result += '\n'
             return result
 79
 80
         def __three_in_row(self, player, start_x, start_y, dx, dy
 81
     ):
             11 11 11
 82
             Determines if a player has three in a row, starting
 83
 84
             from a starting position (start_x, start_y) and going
 85
             in the direction indicated by (dx, dy)
 86
 87
             x = start_x; y = start_y
 88
             for i in range(0,NUM_ROWS):
                  if self.__board[y][x] \neq player:
 89
                      return False
 90
 91
                  x += dx
 92
                  y += dy
 93
             return True
 94
 95
         def __is_winner(self, player):
 96
             """Returns True if and only if the given player has
 97
    won"""
 98
             if self.__three_in_row(player, 0, 0, 1, 1):
 99
100
                  return True
             elif self.__three_in_row(player, 2, 0, -1, 1):
101
                  return True
102
103
             else:
                  for i in range(0, NUM_ROWS):
104
                      if (self.__three_in_row(player, 0, i, 1, 0)
105
                          or self.__three_in_row(player, i, 0, 0, 1
106
     )):
107
                          return True
108
                  return False
109
110
```

def get_winner(self):

111112

File - /Users/chrishegangkim/Desktop/Union College/Spring 2023/CSC 120/Kim_Lab5/tictactoe_board.py

```
113
            Determines if there is a winner and returns the
   player who has won.
114
            :param board: A tictactoe board.
            :return: 'X' if player X is the winner; '0' if player
115
     O is the winner; None if there is no winner.
116
            if self.__is_winner('X'):
117
                return 'X'
118
            elif self.__is_winner('0'):
119
                return '0'
120
121
            else:
122
                return None
123
124
```

```
File - /Users/chrishegangkim/Desktop/Union College/Spring 2023/CSC 120/Kim_Lab5/test_tictactoe_board.py
 1 """
 2 :author: Chris Hegana Kim
 3 :note: I affirm that I have carried out the attached academic
   endeavors with full academic honesty,
 4 in accordance with the Union College Honor Code and the course
     syllabus.
 5 ""
 7 from tictactoe_board import *
 8 from testing import *
 9
10
11 def test_qet_winner():
12
        start_tests("Tests for tictactoe_board.get_winner()")
        test_qet_winner_horiz_X()
13
14
        test_qet_winner_vertical_X()
15
        test_get_winner_diagonal_X_L()
16
        test_get_winner_diagonal_X_R()
17
        test_get_winner_incomplete_board()
18
        test_get_winner_empty()
19
        finish_tests()
20
21 """
22 Individual unit tests start here
23 """
24
25 def test_get_winner_horiz_X():
        a_board = Tictactoe_board(['XXX',
26
27
                                     '00X',
                                     'X00'])
28
        assert_equals(str(a_board) + "Three Xs in a row
29
   horizontally",
30
                       'X',
31
                       a_board.get_winner())
32
33
34 def test_get_winner_vertical_X():
        a_board = Tictactoe_board(['XXO',
35
36
                                     'XOX',
```

37

38

'X00'1)

assert_equals(str(a_board) + "Three Xs in a row vertically

```
File - /Users/chrishegangkim/Desktop/Union College/Spring 2023/CSC 120/Kim Lab5/test tictactoe board.py
38 ",
39
                        'X',
40
                       a_board.qet_winner())
41
42
43 def test_get_winner_diagonal_X_L():
        a_board = Tictactoe_board(['X00',
44
45
                                      'XXO',
                                      'X0X'])
46
        assert_equals(str(a_board) + "Three Xs in a row in the
47
   left diagonal",
48
                        'X',
49
                       a_board.get_winner())
50
51
52 def test_get_winner_diagonal_X_R():
        a_board = Tictactoe_board(['XOX',
53
54
                                      'XXO',
                                      'X00'])
55
56
        assert_equals(str(a_board) + "Three Xs in a row in the
   right diagonal",
57
                        'X',
58
                       a_board.get_winner())
59
60
61 def test_get_winner_incomplete_board():
        a_board = Tictactoe_board(['XXX',
62
63
                                      '00X',
64
                                      'X00'1)
        a_board.clear_cell(0, 0)
65
        assert_equals(str(a_board) + "Incomplete board, no winner
66
   yet",
67
                       None,
                       a_board.get_winner())
68
69
70
71 def test_get_winner_empty():
        a_board = Tictactoe_board(None)
72
73
        assert_equals(str(a_board) + "Empty board, no winner yet",
74
                       None,
75
                       a_board.get_winner())
```

```
File - /Users/chrishegangkim/Desktop/Union College/Spring 2023/CSC 120/Kim_Lab5/test_tictactoe_board.py
```

```
76
77
78 if __name__ = "__main__":
79    test_get_winner()
80
```