```python
from tictactoe_board import *

def main():
    the_board = Tictactoe_board(['XOX',
                                 'OXO',
                                 'XOO'])
    print(the_board)
    print("The winner is %s" % the_board.get_winner())
    print()

    the_board.place_piece(2, 0, 'O')
    print(the_board)
    print("The winner is %s" % the_board.get_winner())

if __name__ == "__main__":
    main()

```

```
"""
Testing utilities.  Do not modify this file!
"""

VERBOSE = True
num_pass = 0
num_fail = 0

def assert_equals(msg, expected, actual):
    """
    Check whether code being tested produces
    the correct result for a specific test
    case. Prints a message indicating whether
    it does.
    :param: msg is a message to print at the beginning.
    :param: expected is the correct result
    :param: actual is the result of the
    code under test.
    """
    if VERBOSE:
        print(msg)

    global num_pass, num_fail

    if expected == actual:
        if VERBOSE:
            print("PASS")
        num_pass += 1
    else:
        if not VERBOSE:
            print(msg)
        print("**** FAIL")
        print("expected: " + str(expected))
        print("actual: " + str(actual))
        if not VERBOSE:
            print("")
        num_fail += 1

```

```
39      if VERBOSE:
40          print("")
41
42
43  def fail_on_error(msg,err):
44      """
45      if run-time error occurs, call this to insta-fail
46
47      :param msg: message saying what is being tested
48      :param err: type of run-time error that occurred
49      """
50      global num_fail
51      print(msg)
52      print("**** FAIL")
53      print(err)
54      print("")
55      num_fail += 1
56
57
58  def start_tests(header):
59      """
60      Initializes summary statistics so we are ready to
    run tests using
61      assert_equals.
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)):
68          print("=",end="")
69      print("")
70      num_pass = 0
71      num_fail = 0
72
73  def finish_tests():
74      """
75      Prints summary statistics after the tests are
```

```
75 complete.
76     """
77     print("Passed %d/%d" % (num_pass, num_pass+
   num_fail))
78     print("Failed %d/%d" % (num_fail, num_pass+
   num_fail))
79     print()
80
```

1. What methods are private?
__row_as_string
__three_in_row
__is_winner

2. What instance variables does it have?
__board: a list that represents the board

3. Write a short description of the internal representation of a board
The board is stored as a 3×3 list of lists of characters. The constructor takes a
list of three 3 character strings and converts them into this 2D structure. If the
inputed Row is none then it will just create an empty board.

```
"""
defines the behavior of a tic-tac-toe board
"""

NUM_ROWS = 3

class Tictactoe_board:

    def __init__(self, rows):
        """
        Constructor. Creates a tictactoe board with given cell values.
        If no initial cell values are given, creates an empty tictactoe board.

        :param rows: A list of three 3-character strings, where each character
        is either 'X', 'O', or ' '.  Each of the
        3-character strings represents a row of the tictactoe board.
        Example: [" X ", "O O", "XXO"] is the board
           | X |
        -----------
         O |   | O
        -----------
         X | X | O
        """
        self.__board = []
        if rows is None:
            empty_row = [' ', ' ', ' ']
            for i in range(NUM_ROWS):
                self.__board.append(empty_row)
        else:
            for i in range(NUM_ROWS):
                row = []
                for j in range(NUM_ROWS):
                    row.append(rows[i][j])
                self.__board.append(row)
```

```
35
36      def place_piece(self, i, j, piece):
37          """
38          Places a piece (either 'X' or 'O') on the board
    .
39
40          :param i: The row in which to place a piece (0
    , 1, or 2)
41          :param j: The column in which to place a piece
     (0, 1, or 2)
42          :param piece: The piece to place ('X' or 'O')
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          """
61          str = ''
62          for column in row[:len(row)-1]:
63              str += column + ' | '
64          str += row[len(row)-1]
65          return str
66
67      def __str__(self):
68          """
69          Produces a string representation of a board,
```

```
 69 returns it.
 70
 71         :return: The string version of the board.
 72         """
 73         result = ''
 74         for row in self.__board[:len(self.__board)-1]:
 75             result += self.__row_as_string(row)
 76             result += '\n----------\n'
 77         result += self.__row_as_string(self.__board[
    len(self.__board)-1])
 78         result += '\n'
 79         return result
 80
 81     def __three_in_row(self, player, start_x, start_y
    , dx, dy):
 82         """
 83         Determines if a player has three in a row,
    starting
 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):
 89             if self.__board[y][x] != player:
 90                 return False
 91             x += dx
 92             y += dy
 93         return True
 94
 95
 96     def __is_winner(self, player):
 97         """Returns True if and only if the given
    player has won"""
 98
 99         if self.__three_in_row(player, 0, 0, 1, 1):
100             return True
101         elif self.__three_in_row(player, 2, 0, -1, 1):
```

```
102             return True
103         else:
104             for i in range(0, NUM_ROWS):
105                 if (self.__three_in_row(player, 0, i,
    1, 0)
106                     or self.__three_in_row(player, i,
    0, 0, 1)):
107                     return True
108             return False
109 
110 
111     def get_winner(self):
112         """
113         Determines if there is a winner and returns
    the player who has won.
114         :param board: A tictactoe board.
115         :return: 'X' if player X is the winner; 'O' if
     player O is the winner; None if there is no winner.
116         """
117         if self.__is_winner('X'):
118             return 'X'
119         elif self.__is_winner('O'):
120             return 'O'
121         else:
122             return None
123 
124 
```

```
"""
:author: James Lin
"""

from tictactoe_board import *
from testing import *


def test_get_winner():
    start_tests("Tests for tictactoe_board.get_winner()")

    test_get_winner_horiz_X()
    test_get_winner_horiz_mid()
    test_get_winner_horiz_O()
    test_get_winner_vertical_O()
    test_get_winner_incomplete_board()
    test_get_winner_draw()
    test_get_winner_empty()
    test_get_winner_diagnol_O()
    test_get_winner_diagnol_X()
    test_get_winner_diagonal()

    finish_tests()

"""
Individual unit tests start here
"""

def test_get_winner_horiz_X():
    a_board = Tictactoe_board(['XXX',
                               'OOX',
                               'XOO'])
    assert_equals(str(a_board) + "Three Xs in a row horizontally",
                  'X',
                  a_board.get_winner())

```

```
37 def test_get_winner_horiz_O():
38     a_board = Tictactoe_board(['XOX',
39                                'XXO',
40                                'OOO'])
41     assert_equals(str(a_board) + "Three Os in a row
   horizontally",
42                   'O',
43                   a_board.get_winner())
44
45 def test_get_winner_horiz_mid():
46     a_board = Tictactoe_board(['XOX',
47                                'OOO',
48                                'XOX'])
49     assert_equals(str(a_board) + "Three Os in a row
   horizontally",
50                   'O',
51                   a_board.get_winner())
52
53 def test_get_winner_vertical_O():
54     a_board = Tictactoe_board(['OOX',
55                                'OXO',
56                                'OXX'])
57     assert_equals(str(a_board) + "Three Os in a row
   vertically",'O', a_board.get_winner())
58
59
60 def test_get_winner_incomplete_board():
61     a_board = Tictactoe_board(['XXX',
62                                'OOX',
63                                'XOO'])
64     a_board.clear_cell(0, 0)
65     assert_equals(str(a_board) + "Incomplete board, no
   winner yet",
66                   None,
67                   a_board.get_winner())
68
69
70 def test_get_winner_draw():
```

```
 71     a_board = Tictactoe_board(['XOX',
 72                                'OXO',
 73                                'OXX'])
 74     assert_equals(str(a_board) + "Full board, draw (no
    winner)",
 75                   None,
 76                   a_board.get_winner())
 77
 78
 79 def test_get_winner_empty():
 80     a_board = Tictactoe_board(None)
 81     assert_equals(str(a_board) + "Empty board, no
    winner yet",
 82                   None,
 83                   a_board.get_winner())
 84
 85 def test_get_winner_diagnol_X():
 86     a_board = Tictactoe_board(['XOX',
 87                                'OXX',
 88                                'XOO'])
 89     assert_equals(str(a_board) + "Diagnol, X wins ", '
    X', a_board.get_winner())
 90
 91 def test_get_winner_diagnol_O():
 92     a_board = Tictactoe_board(['XOO',
 93                                'XOX',
 94                                'OXX'])
 95     assert_equals(str(a_board) + "Diagnol, O wins ", '
    O', a_board.get_winner())
 96
 97 def test_get_winner_diagonal():
 98     a_board = Tictactoe_board(['OXX',
 99                                'XOX',
100                                'XXO'])
101     assert_equals(str(a_board) + "Diagonal", 'O',
    a_board.get_winner())
102
103 if __name__ == "__main__":
```

```
104     test_get_winner()
105
```