```
"""
functions related to creating, printing,
and evaluating tic-tac-toe boards

:author: James Lin
:note: I affirm that I have carried out the attached
academic endeavors with full academic honesty, in
accordance with the Union College Honor Code and the
course syllabus.
"""


def remove_blank_lines(list_of_strings):
    """
    Given a list of strings, return a copy
    with all empty strings removed
    :param list_of_strings: list of strings, some of
which may be ''; this list is unchanged
    :return: list identical to list_of_strings, but all
 empty strings removed
    """
    result = list()
    for s in list_of_strings:
        if s != '':
            result.append(s)
    return result


def get_board_from_file(filename):
    """
    Reads board, returns a list of rows.
    :param filename: text file with a tic-tac-toe board
 such as
    X X X
    O X O
    X O O
    where each line is one row
    :return: list of strings where each string is a
```

```
33      row from filename; any blank lines in the file are
    removed
34      Example: ["X X X", "O X O", "X O O"]
35      """
36      board_list = []
37      board_file = open(filename, "r")
38      for line in board_file:
39          board_list.append(line.strip())
40      board_file.close()
41      board_list = remove_blank_lines(board_list)
42      return board_list
43
44
45  def print_row(row):
46      """
47      Nicely prints a row of the board.
48      :param row: string of Xs and Os
49      """
50      nice_row = ''
51      for i in range(0, len(row)):
52          nice_row += row[i]
53          if i != len(row) - 1:
54              nice_row += ' | '
55      print(nice_row)
56
57
58  def print_board(board):
59      """
60      prints the tic-tac-toe board
61      :param board: list of rows
62      """
63      for i in range(0, len(board)):
64          row = board[i]
65          print_row(row)
66          if i != len(board) - 1:
67              print('----------')
68
69
```

```
70 def three_in_row(board, player, start_x, start_y, dx,
   dy):
71     """
72     Determines if a player has three in a row,
   starting
73     from a starting position (start_x, start_y) and
   going
74     in the direction indicated by (dx, dy). Example:
75     (start_x, start_y) = (2,2) means we start at the
   lower
76     right (row 2, col 2). (dx, dy) = (-1, 0) means the
    next
77     square we check is (2+dx, 2+dy) = (1,2).  And the
   last
78     square we check is (1+dx, 2+dy) = (0,2).  So we've
    just
79     checked the rightmost column - (2,2), (1,2), and (
   0,2).
80     :param board: list of rows
81     :param player: string -- either "X" or "O"
82     :param start_x: row to start checking at; first
   row is row 0
83     :param start_y: col to start checking at; first
   col is col 0
84     :param dx: 1 if checking downward, -1 if checking
   upward, 0 if checking this row
85     :param dy: 1 if checking rightward, -1 if checking
    leftward, 0 if checking this col
86     """
87     x = start_x
88     y = start_y
89     for i in range(0, 3):
90         if board[x][y] != player:
91             return False
92         x += dx
93         y += dy
94     return True
95
```

```
 96
 97 def is_winner(board, player):
 98     """
 99     Returns True if and only if the given player has
    won.
100     :param board: list of row strings
101     :param player: string - "X" or "O"
102     :return: True if player won; False if player lost
    or tied
103     """
104     if three_in_row(board, player, 0, 0, 1, 1) or
    three_in_row(board, player, 2,0, -1, 1):
105         return True
106     else:
107         for i in range(0, 3):
108             if (three_in_row(board, player, 0, i, 1, 0
    )
109                     or three_in_row(board, player, i,
    0, 0, 1)):
110                 return True
111         return False
112
113
114 def get_winner(board):
115     """
116     Returns the name of the winner, or None if there
    is no winner
117     :param board: list of row strings
118     :return: "X" if X is winner, "O" if O is winner,
    None if tie
119     """
120     if is_winner(board, 'X'):
121         return 'X'
122     elif is_winner(board, 'O'):
123         return 'O'
124     else:
125         return None
126
```

```
127
128 def confirm_result(board, winner):
129     """
130     Checks if the actual result is the same as the
    expected result and prints the result.
131     :param board: list of row strings
132     :param winner: the expected result for the
    tictactoe
133     """
134     if get_winner(board) == winner:
135         print(f"PASS: {winner} ")
136     else:
137         print(f"FAIL: Expected {winner} but got {
    get_winner(board)}")
138
139
140 def main():
141     board = get_board_from_file("X_wins.txt")
142     print_board(board)
143     confirm_result(board, 'X')
144
145     board1 = get_board_from_file("O_wins.txt")
146     print_board(board1)
147     confirm_result(board1, 'O')
148
149     board2 = get_board_from_file("None_wins.txt")
150     print_board(board2)
151     confirm_result(board2, None)
152
153 def main2():
154     board1 = ["XXX",
155               "OOX",
156               "XOO"]
157     print_board(board1)
158     confirm_result(board1, 'X')
159
160     board2 = ["XXO",
161               "XOX",
```

```
162                 "OOX"]
163     print_board(board2)
164     confirm_result(board2, 'O')
165
166     board3 = ["XXO",
167               "OOX",
168               "XOX"]
169     print_board(board3)
170     confirm_result(board3, None)
171
172
173 if __name__ == "__main__":
174     main2()
175
```