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

:author: Neil Daterao
: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
    row from filename; any blank lines in the file are removed
    Example: ["X X X", "O X O", "X O O"]
```

```
36      """
37      board_list = []
38      board_file = open(filename, "r")
39      for line in board_file:
40          board_list.append(line.strip())
41      board_file.close()
42      board_list = remove_blank_lines(board_list)
43      return board_list
44
45
46  def print_row(row):
47      """
48      Nicely prints a row of the board.
49      :param row: string of Xs and Os
50      """
51      nice_row = ''
52      for i in range(0, len(row)):
53          nice_row += row[i]
54          if i != len(row) - 1:
55              nice_row += ' | '
56      print(nice_row)
57
58
59  def print_board(board):
60      """
61      prints the tic-tac-toe board
62      :param board: list of rows
63      """
64      for i in range(0, len(board)):
65          row = board[i]
66          print_row(row)
67          if i != len(board) - 1:
68              print('----------')
69
70
71  def three_in_row(board, player, start_x, start_y, dx
    , dy):
72      """
73      Determines if a player has three in a row,
    starting
74      from a starting position (start_x, start_y) and
```

```
 74 going
 75     in the direction indicated by (dx, dy). Example:
 76     (start_x, start_y) = (2,2) means we start at the
     lower
 77     right (row 2, col 2). (dx, dy) = (-1, 0) means
    the next
 78     square we check is (2+dx, 2+dy) = (1,2).  And
    the last
 79     square we check is (1+dx, 2+dy) = (0,2).  So we'
    ve just
 80     checked the rightmost column - (2,2), (1,2), and
     (0,2).
 81     :param board: list of rows
 82     :param player: string -- either "X" or "O"
 83     :param start_x: row to start checking at; first
    row is row 0
 84     :param start_y: col to start checking at; first
    col is col 0
 85     :param dx: 1 if checking downward, -1 if
    checking upward, 0 if checking this row
 86     :param dy: 1 if checking rightward, -1 if
    checking leftward, 0 if checking this col
 87     """
 88     x = start_x
 89     y = start_y
 90     for i in range(0, 3):
 91         if board[x][y] != player:
 92             return False
 93         x += dx
 94         y += dy
 95     return True
 96
 97
 98 def is_winner(board, player):
 99     """
100     Returns True if and only if the given player has
     won.
101     :param board: list of row strings
102     :param player: string - "X" or "O"
103     :return: True if player won; False if player
    lost or tied
```

```
104     """
105     if three_in_row(board, player, 0, 0, 1, 1) or
    three_in_row(board, player, 0, 2, 1, -1):
106         return True
107     else:
108         for i in range(0, 3):
109             if (three_in_row(board, player, 0, i, 1
    , 0)
110                     or three_in_row(board, player, i
    , 0, 0, 1)):
111 
112                 return True
113         return False
114 
115 
116 def get_winner(board):
117     """
118     Returns the name of the winner, or None if there
     is no winner
119     :param board: list of row strings
120     :return: "X" if X is winner, "O" if O is winner
    , None if tie
121     """
122     if is_winner(board, 'X'):
123         return 'X'
124     elif is_winner(board, 'O'):
125         return 'O'
126     else:
127         return None
128 
129 def confirm_result(board, expected_winner):
130     """
131     Tests if the winner of the tic-tae-toe game is
    supposed to be the actual winner. If so, function
    will print PASS.
132     If not, the function will print FAIL accompanied
     by the failed test case.
133 
134 
135     :param board: Takes tic-tac-toe board as a
    string
```

```
136     :param expected_winner: Expected winning player
    of tic tac toe game
137     :return: None
138     """
139
140     actual_winner = get_winner(board)
141     if actual_winner == expected_winner:
142         print("PASS")
143     else:
144         print("FAIL")
145         print_board(board)
146         print("Actual Winner: %s wins" %
    actual_winner)
147         print("Expected Winner: %s wins" %
    expected_winner)
148
149
150 def main():
151     """
152     Reads boards from test case files and tests them
153     :return: None
154     """
155     test_cases = [("X_wins.txt","X"), ("O_wins.txt"
    , "O"), ("X_diag_wins_left2right.txt","X"), ("
    X_diag_wins_right2left.txt","X"), ("
    O_diag_wins_right2left.txt","O"), ("
    O_diag_wins_left2right.txt", "O"), ("Draw.txt", None
    )]
156     for test in test_cases:
157         board = get_board_from_file(test[0])
158         expected_winner = test[1]
159         confirm_result(board, expected_winner)
160
161
162 def main2():
163     """
164     Runs hard-coded test cases
165
166     :return: None
167     """
168     x_wins = ["XXX",
```

```
169                 "OOX",
170                 "XOO"]
171     confirm_result(x_wins, "X")
172 
173     o_wins = ["OXX",
174               "OOO",
175               "XXO"]
176     confirm_result(o_wins, "O")
177 
178     x_diag_wins_left2right = ["XOX",
179                               "OXO",
180                               "OOX"]
181     confirm_result(x_diag_wins_left2right, "X")
182 
183     x_diag_wins_right2left= ["XOX",
184                              "OXO",
185                              "XOO"]
186     confirm_result(x_diag_wins_right2left, "X")
187 
188     o_diag_wins_right2left = ["OXO",
189                               "XOX",
190                               "OXX" ]
191     confirm_result(o_diag_wins_right2left, "O")
192 
193     o_diag_wins_left2right = ["OXO",
194                               "XOX",
195                               "XXO"]
196     confirm_result(o_diag_wins_left2right, "O")
197 
198     draw = ["OOX",
199             "XXO",
200             "OXO"]
201     confirm_result(draw, None)
202 
203 
204 
205 if __name__ == "__main__":
206     main()
207     print("\nMain2:")
208     main2()
209 
```