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

:author: Chris Hegang Kim
: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"]
    """
```

```
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 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 # 0
89      y = start_y # 0
90      for i in range(0, 3):
91          if board[x][y] ≠ player:
92              return False
93          x += dx # 1
94          y += dy # 1
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, 2 , 0, -1, 1): # Detect both left
     and right diagonals
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                 return True
```

```
112         return False
113 
114 
115 def get_winner(board):
116     """
117     Returns the name of the winner, or None if there is no
    winner
118     :param board: list of row strings
119     :return: "X" if X is winner, "O" if O is winner, None if
    None_wins
120     """
121     if is_winner(board, 'X'):
122         return 'X'
123     elif is_winner(board, 'O'):
124         return 'O'
125     else:
126         return None
127 
128 
129 def confirm_result(board, expected_winner):
130     """
131     Compares the expected winner and the actual result
132     :param board: a list of string rows
133     :param expected_winner: a string for the expected winner
    or None if it is a tie
134     :return:
135     """
136     actual_result = get_winner(board)
137 
138     if actual_result == expected_winner:
139         print("PASS")
140 
141     else:
142         print("FAIL")
143         print_board(board)
144         print("actual result: ", actual_result, ", expected
    winner: ", expected_winner)
145 
146 
147 def main():
148     """
```

```
149     Tests input files with different board states
150     :return:
151     """
152     board = get_board_from_file("first_row_X_wins.txt")
153     confirm_result(board, "X")
154
155     board = get_board_from_file("right_diagonal_X_wins")
156     confirm_result(board, "X")
157
158     board = get_board_from_file("None_wins")
159     confirm_result(board, None)
160
161     board = get_board_from_file("first_column_X_wins")
162     confirm_result(board, "X")
163
164
165 def main2():
166     """
167     Tests hard-codes with different board states
168     :return:
169     """
170     board = ["XXX",
171              "OOX",
172              "XOO"]
173     confirm_result(board, "X")
174
175     board = ["XOX",
176              "OXX",
177              "XOO"]
178     confirm_result(board, "X")
179
180     board = ["XXO",
181              "OOX",
182              "XOX"]
183     confirm_result(board, None)
184
185     board = ["XOO",
186              "XOX",
187              "XXO"]
188     confirm_result(board, "X")
189
```

```
190
191 if __name__ == "__main__":
192     main()
193     main2()
```