Міністерство освіти і науки України
Національний авіаційний університет
Навчально-науковий інститут комп'ютерних інформаційних технологій
Кафедра комп'ютеризованих систем управління

Лабораторна робота №2
з дисципліни «Системне програмування»
на тему «Функції»

Виконав:
студент ННІКІТ СП-225
Клокун Владислав
Перевірив:
Радченко П. В.

Київ 2017

1 Завдання

Створити гру «Хрестики-нулики» для командного рядка з підтримкою покрокової гри двох гравців.

2 Розв'язання

```
PLAYER_O = 'O'
PLAYER_X = 'X'

ROWS = 3
COLS = 3

class Error(Exception):
    """Base class for exceptions"""
    pass

class MovementError(Error):
    """ Inherits from Error """
    pass

class Board:
    def __init__(self):
        # Create an empty board.
        self.board = [
            [' ', ' ', ' '],
            [' ', ' ', ' '],
            [' ', ' ', ' ']
        ]

        self.row_count = ROWS
        self.col_count = COLS

    def show(self):
        for line in self.board:
            print(line)

    def is_full(self):
        for line in self.board:
            # If there is a space on a board, it's not full.
            if ' ' in line:
                return False
```

```
36
37          return True
38
39      def columns(self):
40          # Return board "as is"
41          return self.board
42
43      def rows(self):
44          # Return transposed board
45          return [list(x) for x in zip(*self.board)]
46
47      def main_diagonal(self):
48          return [self.board[0][0], self.board[1][1], self.
            board[2][2]]
49
50      def anti_diagonal(self):
51          return [self.board[0][2], self.board[1][1], self.
            board[2][0]]
52
53      def set_position(self, pos_x, pos_y, val):
54          if not self.position_is_valid(pos_x, pos_y):
55              raise MovementError('position not valid')
56
57          self.board[pos_x][pos_y] = val
58
59      def position_is_valid(self, pos_x, pos_y):
60          try:
61              curval = self.board[pos_x][pos_y] != ' '
62          except Exception as e:
63              print('Cannot retrieve board value: {}'.format(
                e))
64              return False
65
66          if self.board[pos_x][pos_y] != ' ':
67              return False
68
69          if pos_x > self.col_count or pos_y > self.row_count
            :
70              return False
71
72          return True
73
74  class TicTacToeGame:
```

```
75      def __init__(self):
76          # Create an empty board
77          self.board = Board()
78
79      def show_board(self):
80          self.board.show()
81
82      def move(self, player, pos_x, pos_y):
83          try:
84              self.board.set_position(pos_x, pos_y, player)
85          except Exception as e:
86              raise e
87
88      def get_winner(self):
89          for player in (PLAYER_X, PLAYER_O):
90              if line_wins(self.board.main_diagonal(), player
                ):
91                  return player
92
93              if line_wins(self.board.anti_diagonal(), player
                ):
94                  return player
95
96              for line in self.board.rows():
97                  if line_wins(line, player):
98                      return player
99
100             for line in self.board.columns():
101                 if line_wins(line, player):
102                     return player
103
104     def is_over(self):
105         if self.board.is_full() or self.get_winner():
106             return True
107
108         return False
109
110 def player_make_move(game, player):
111     while True:
112         try:
113             x, y = map(int, input('Player ' + player
114             + ', please make a move (x, y): '))
115             game.move(player, x, y)
```

```
            break
        except Exception as e:
            print('There was an error: {}'.format(e))

def main():
    print('This is a tic-tac-toe game.\n'
          'Cells are 0-indexed as (x, y).\n')

    tictactoe = TicTacToeGame()

    tictactoe.board.show()

    current_player = 0
    while not tictactoe.is_over():
        players = (PLAYER_X, PLAYER_O)
        player_make_move(tictactoe, players[current_player
        ])

        current_player = (current_player + 1) % 2

        tictactoe.board.show()
        print('\n')

    winner = tictactoe.get_winner()
    if winner != None:
        print('Player ' + winner + ' won the game!')
    else:
        print('Game ended in a draw.')

# Checks if line wins the game for a player.
def line_wins(line, player):
    return line == list(player * len(line))

if __name__ == '__main__':
    main()
```