Kő papír olló

A játék lényege:

A játékosok hangosan a kő – papír – olló-t mondják, minden szónál megemelve az ökölbe szorított kezüket. A harmadik szó után, vagy az utána következő ütemben a játékosok kezükkel felveszik a három alakzat egyikét és megmutatják az ellenfelüknek.

Jelzések:
– kő: zárt ököl
– papír: nyitott tenyér
– olló: kinyújtott, szétnyitott mutató és középső ujj

A cél az, hogy olyat mutassunk, amely legyőzi a másikat:
– a kő kicsorbítja az ollót: a kő győz
– az olló elvágja a papírt: az olló győz
– a papír becsomagolja a követ: a papír győz
– Ha mindketten ugyanazt mutatják, a játék döntetlen és újat játszanak.

```
import math

import time

from player import HumanPlayer, RandomComputerPlayer, SmartComputerPlayer



class TicTacToe():

    def __init__(self):

        self.board = self.make_board()

        self.current_winner = None


        def make_board():

        return [' ' for _ in range(9)]


    def print_board(self):

        for row in [self.board[i*3:(i+1) * 3] for i in range(3)]:

            print('| ' + ' | '.join(row) + ' |')
```

```
  def print_board_nums():
    # 0 | 1 | 2
    number_board = [[str(i) for i in range(j*3, (j+1)*3)] for j in range(3)]
    for row in number_board:
        print('| ' + ' | '.join(row) + ' |')


def make_move(self, square, letter):
    if self.board[square] == ' ':
        self.board[square] = letter
        if self.winner(square, letter):
            self.current_winner = letter
        return True
    return False


def winner(self, square, letter):
    # check the row
    row_ind = math.floor(square / 3)
    row = self.board[row_ind*3:(row_ind+1)*3]
    # print('row', row)
    if all([s == letter for s in row]):
        return True
    col_ind = square % 3
    column = [self.board[col_ind+i*3] for i in range(3)]
    # print('col', column)
    if all([s == letter for s in column]):
        return True
    if square % 2 == 0:
        diagonal1 = [self.board[i] for i in [0, 4, 8]]
```

```
            # print('diag1', diagonal1)
            if all([s == letter for s in diagonal1]):
                return True
            diagonal2 = [self.board[i] for i in [2, 4, 6]]
            # print('diag2', diagonal2)
            if all([s == letter for s in diagonal2]):
                return True
        return False

    def empty_squares(self):
        return ' ' in self.board

    def num_empty_squares(self):
        return self.board.count(' ')

    def available_moves(self):
        return [i for i, x in enumerate(self.board) if x == " "]


def play(game, x_player, o_player, print_game=True):

    if print_game:
        game.print_board_nums()

    letter = 'X'
    while game.empty_squares():
        if letter == 'O':
            square = o_player.get_move(game)
        else:
```

```
            square = x_player.get_move(game)
        if game.make_move(square, letter):

            if print_game:
                print(letter + ' makes a move to square {}'.format(square))
                game.print_board()
                print('')

            if game.current_winner:
                if print_game:
                    print(letter + ' wins!')
                return letter  # ends the loop and exits the game
            letter = 'O' if letter == 'X' else 'X'  # switches player

        time.sleep(.8)

    if print_game:
        print('It\'s a tie!')



if __name__ == '__main__':
    x_player = SmartComputerPlayer('X')
    o_player = HumanPlayer('O')
    t = TicTacToe()
    play(t, x_player, o_player, print_game=True)
```