Всё же я решил использовать двумерный массив для работы с доской. Но при этом создать доску 9\*9, чтобы не было 0 линии. Также надо перевернуть массив. Чтобы у нас были нормальные координаты.

Представим себе начальную доску. Значит, чтобы переназначить координаты нам надо каждой координате передать значение 8-х и 8-у.

Но так не очень получается. Поэтому придётся оставить координаты как есть.

Итак, мы взяли за основу принцип реализации доски из видео. Больше ничего оттуда я брать не буду.

Теперь необходимо написать для всех фигур правила их ходов. В особенности сложным будет правило рокировки, а также взятие на проходе и перекрытие от шаха, а также связки на фигуру и короля.

class King(object):

def \_\_init\_\_(self, color, x, y):

self.color = color

self.x = x

self.y = y

def \_\_str\_\_(self):

if (self.color == 'White'):

return 'K'

else:

if (self.color == 'Black'):

return 'k'

def Moves(self):

moves = []

if (self.color == 'White'):

if(0 <= self.x - 1 <= 7 and 0 <= self.y - 1 <= 7 and Board[self.x - 1][self.y - 1] == '.'):

moves.append([self.x - 1, self.y - 1])

if(0 <= self.y - 1 <= 7 and Board[self.x][self.y - 1] == '.'):

moves.append([self.x, self.y - 1])

if(0 <= self.x + 1 <= 7 and 0 <= self.y - 1 <= 7 and Board[self.x + 1][self.y - 1] == '.'):

moves.append([self.x + 1, self.y - 1])

if(0 <= self.x - 1 <= 7 and Board[self.x - 1][self.y] == '.'):

moves.append([self.x - 1, self.y])

if(0 <= self.x + 1 <= 7 and Board[self.x + 1][self.y] == '.'):

moves.append([self.x + 1, self.y])

if(0 <= self.x - 1 <= 7 and 0 <= self.y + 1 <= 7 and Board[self.x - 1][self.y + 1] == '.'):

moves.append([self.x - 1, self.y + 1])

if(0 <= self.y + 1 <= 7 and Board[self.x][self.y + 1] == '.'):

moves.append([self.x, self.y + 1])

if(0 <= self.x + 1 <= 7 and 0 <= self.y + 1 <= 7 and Board[self.x + 1][self.y + 1] == '.'):

moves.append([self.x + 1, self.y + 1])

return moves

Board = [[],[],[],[],[],[],[],[]]

for i in range(8):

Board[i] = ['.', '.', '.', '.', '.', '.', '.', '.']

Board[7][4] = King('White', 7, 4)

Print\_Board = ''

for y in range(8):

Print\_Board += ''.join(map(str, Board[y])) + "\n"

print(Print\_Board)

print(King('White', 7, 4).Moves())

Теперь хорошо бы заменить цвет на цифру, чтобы писать ходы короля вне зависимости от цвета. То есть ходы будут одинаковые, но проверка на фигуру противника будет, как 1-self.color

Мне пришлось отказаться от этой идеи, поскольку я не придумал, как реализовать её.

Зато мне удалось соединить методы Андрея, мужика и свои в единый код.

Сейчас я пишу программу для слона. Уже написал для коня, пешки и короля. Единственная проблема заключается в том, что я пока что не могу делать взятия, рокировку, брать на проходе, получать шахи и мат, превращать пешку в ферзи (и не только в ферзи).

Это можно будет реализовать позже.

Пока что мне нужно создать все фигуры.

Особые правила можно будет написать позже. Как и сам перебор всех возможных ходов в позиции.

ТАКЖЕ ПОТОМ МОЖНО БУДЕТ УЛУЧШИТЬ КОД. ПРИВЕСТИ ФИГУРЫ К ОДНОМУ КЛАССУ. И ЕСЛИ НЕ РЕАЛИЗУЮ СЕЙЧАС, РЕАЛИЗОВАТЬ ПЕРЕБОР ВОЗМОЖНЫХ ХОДОВ СЛОНОМ ЧЕРЕЗ WHILE.

ТАКЖЕ ДЛЯ СЛОНА НЕ НЕОБХОДИМО ВЕРХНЕЕ ОГРАНИЧЕНИЕ В ПРОВЕРКЕ ВОЗМОЖНЫХ ХОДОВ.