Chess game

Im using pygame for making this. Before we begin, you need to install the Pygame library. Open a terminal (or command prompt) and run the following command:

pip install pygame

Or my code won’t work xD

1. Making a board for the game
   1. I will use pygame for making game UI, make a white screen, using board() to loop, i%4 to make the columns position, i//4 to make the rows. My screen is 500x500 so 50 pixels for the squares should be good.
2. Pieces image and location
   1. k = king / q = queen / rook = r / bigshop = b / knight = n / pawn = p
   2. Create 2 types of var, w\_pieces (white pieces) and w\_cord (white locations)
   3. Black is same with white but replace w with b
   4. Import images for each type of pieces, put them all together
   5. Blit the images at the cordness of pieces, resize it to 50x50 to fit with the square
   6. Cordness of pieces = w\_cord/b\_cord \* 50
3. Piece selection
   1. When on click, x and y pos will be // 50 and put then in click, so “click” should be the cordness same with w\_cord and b\_cord
   2. For the turn of white and black, I use turn\_step, if turn\_step <= 1 its white turn, if turn\_step >1 its black turn. Why I don’t use turn\_step == 0 for white and turn\_step == 1 for black, because player can choose another piece if they click on wrong one
   3. Explain about “selection”:
      1. Selection is same as the id number of the pieces, they help the rect I drew to get the cord and draw at the piece’s cordness. Ex: slection = 4 so we can know its black king and the cordness is (4, 0)
   4. Check if the selection piece in valid\_moves (below):
      1. w\_cord[selection] = click (move the piece by changing its cordness in w\_cord with click
      2. If click on b\_cord (their enemy):
      3. b\_piece to save its id, and pop them (delete it)
      4. Check option (below)
4. Valid moves for the pieces
   1. Check type of piece and run the relative def
   2. Take the valid moves values (form check possible move from the below funtions) and append in possible moves
5. Valid moves of them
   1. Pawn:  
      Check forward 1 and 2 squares, if position + 1 not in b\_cord (an enemy forward) and not outside the board, append moves. If there is an enemy up beside the pawn, can capture them
   2. Rook:  
      Check in 2 straight paths, if there is an enemy, stop the path, append the valid moves.
   3. Bigshop:  
      Check in 2 cross paths, if there is an enemy, stop the path, append the valid moves.
   4. Knight:  
      8 squares to check for knights, they can go two squares in one direction and one in another, append the valid moves.
   5. Queen:  
      Queen movement is basicly same with bigshop and rook, so I compare pos\_b (check bigshop) and pos\_r (check rook).
   6. King:  
      8 squares to check for kings, they can go one square any direction, change depending on their location in the board.
6. UI improvements (if I have time)
   1. Draw valid moves with black circle, using valid\_moves I got in check\_options()
   2. Black and white score, restart button