Основные объекты и структуры данных

1. Фигура (piece)

```
# Структура фигуры
piece = {
    'symbol': 'P',      # тип фигуры: P, R, N, B, Q, K
    'color': 'white'    # цвет: 'white' или 'black'
}
```

2. Доска (board)

```
# Двумерный список 8x8
board = [
    [None, None, None, ...],  # строка 0
    [None, None, None, ...],  # строка 1
    # ... 8 строк
]
# Индексы: board[y][x], где y - строка (0-7), x - столбец (0-7)
```

3. Состояние игры (state)

```
python
state = {
    'board': board,          # текущее состояние доски
    'history': [],           # список строк с историей ходов
    'turn': 'white'          # чей сейчас ход
}
```

Взаимосвязи объектов

```
state -> содержит -> board -> содержит -> pieces
                  \
                   -> history (записи о перемещениях фигур)
```

Последовательность выполнения

Фаза 1: Инициализация

play() -> init_board() -> создание фигур -> заполнение доски

Фаза 2: Игровой цикл

```
while True:
    print_board()          # отрисовать доску
    print_history()        # показать историю
    check_checkmate()      # проверить мат
    check_check()          # проверить шах

    if turn == 'white':
        user_turn()        # ход игрока
    else:
        bot_turn()         # ход бота
```

Детализация ключевых функций

get_moves(piece, position, board)

```
# Вход: фигура, её позиция (x,y), доска
# Выход: список допустимых ходов [(x1,y1), (x2,y2), ...]
```

Алгоритм:
1. Определить тип фигуры (symbol)
2. Для каждого типа - свой алгоритм генерации ходов:
   - Пешка: вперед, взятие по диагонали
   - Конь: 8 возможных L-образных ходов
   - Король: 8 соседних клеток
   - Слон/Ладья/Ферзь: linear_moves()

linear_moves(position, board, directions)

```
# Обрабатывает линейное движение (слон, ладья, ферзь)
```

 # directions: список направлений [(dx,dy), ...]

Для каждого направления:
  Двигаться по прямой, пока не встретим:
  - Пустую клетку -> добавить ход
  - Чужую фигуру -> добавить ход и остановиться
  - Свою фигуру -> остановиться

move_piece(state, from_pos, to_pos)

Алгоритм:
1. Извлечь фигуру из from_pos
2. Записать ход в history в формате:
  "(время)Фигура откуда -> куда (взятие если было)"
3. Переместить фигуру на to_pos
4. Очистить from_pos
5. Сменить ход (turn)

get_all_valid_moves(board, color)

 # Все допустимые ходы для цвета с учетом шаха

Алгоритм:
1. Для каждой фигуры цвета на доске:
2. Получить все возможные ходы (get_moves)
3. Для каждого хода создать копию доски
4. Проверить: не оставляет ли ход короля под шахом?
5. Если нет - добавить в допустимые ходы

Взаимодействие объектов в типичном ходе

Ход игрока:

user_turn() -> parse input -> validate -> move_piece()
↓
get_all_valid_moves() -> get_moves() -> linear_moves()
|

in_check() (проверка шаха)

Ход бота:

bot_turn() -> get_all_valid_moves() -> random.choice() -> move_piece()

Система истории

Формат записи:
(время)Фигура откуда -> куда (взятие)

Пример: (14:30:25)P e2 -> e4

Сохранение: save_history() записывает в файл game_history.txt

Проверка правил

Шах: in_check() - король атакован фигурой противника

Мат: is_checkmate() - шах + нет допустимых ходов

Валидация хода: проверка что:

- Ход существует в get_all_valid_moves()
- Фигура принадлежит игроку
- Координаты в пределах доски

Поток данных

Пользовательский ввод -> Валидация -> Изменение state -> Обновление доски
Бот -> Выбор хода -> Изменение state -> Обновление доски

Основные компоненты

1. Инициализация игры

```
def play():
    state = {
        'board': init_board(),  # создание доски
        'history': [],          # история ходов
        'turn': 'white'         # чей ход (начинают белые)
    }
```

2. Создание доски (init_board())

- Создает двумерный список 8×8
- Расставляет фигуры в начальной позиции:
    - Пешки: строки 1 (черные) и 6 (белые)
    - Фигуры: строки 0 (черные) и 7 (белые)
    - Порядок фигур: Ладья, Конь, Слон, Ферзь, Король, Слон, Конь, Ладья

3. Отображение доски (print_board(board))

- Выводит доску с буквенными (a-h) и цифровыми (1-8) координатами
- Белые фигуры отображаются заглавными буквами, черные - строчными

4. Определение возможных ходов (get_moves(piece, position, board))

Для каждой фигуры свой алгоритм:

- **Пешка (P)**: движение вперед, взятие по диагонали, двойной ход в начале
- **Конь (N)**: ход буквой "Г" в 8 направлениях
- **Король (K)**: ход на 1 клетку в любом направлении
- **Слон (B)**: движение по диагонали (linear_moves)
- **Ладья (R)**: движение по вертикали/горизонтали (linear_moves)
- **Ферзь (Q)**: комбинация слона и ладьи (linear_moves)

5. Проверка шахов и матов

- in_check(board, color): проверяет, находится ли король под шахом
- is_checkmate(board, color): проверяет мат (шах + нет допустимых ходов)
- get_all_valid_moves(board, color): все допустимые ходы с учетом шаха

6. Ход игрока (user_turn(state))

- Запрос ввода хода (формат: "e2 e4")
- Проверка корректности хода
- Выполнение хода через move_piece()

7. Ход бота (bot_turn(state))

- Выбирает случайный ход из всех допустимых
- Выполняет ход через move_piece()

8. Механика хода (move_piece(state, from_pos, to_pos))

- Перемещает фигуру на доске
- Добавляет запись в историю ходов
- Меняет активного игрока

9. Сохранение истории (save_history(history))

- Записывает историю игры в файл game_history.txt

Последовательность работы

1. Инициализация доски и состояния игры
2. Цикл игры до мата или пата
3. Отображение доски и истории
4. Проверка шаха/мата
5. Ход игрока (белые) или бота (черные)
6. Сохранение истории при завершении игры