Improving in Chess using Python

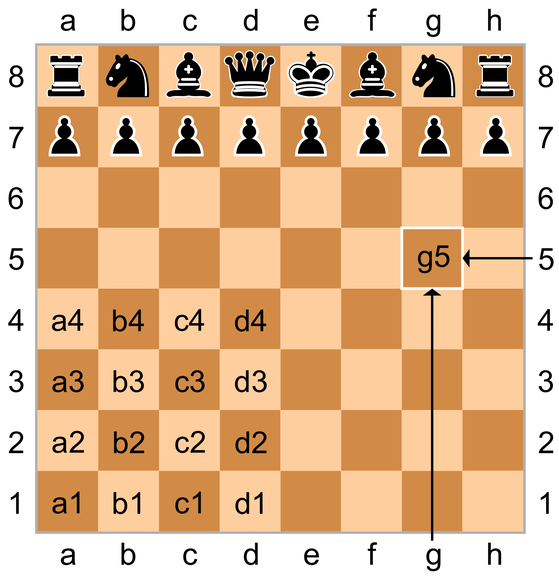
Adarsh Divakaran | adarsh@lexoga.com

Algebraic notation

Algebraic notation is the standard method for recording individual chess moves using coordinates based on the 8×8 grid (files a–h and ranks 1–8). Each move is described by the piece's abbreviation (omitted for pawns) and the destination square, with special symbols for captures (x), checks (+), and checkmate (#).

For example, Nf3 means a knight moved to f3, and exd5 means a pawn captured on d5.

Piece	Symbol	Example
Pawn	*none*	e4, d5
Knight	N	Nc3
Bishop	B	Be6
Rook	R	Re1
Queen	Q	Qd2
King	K	Ke1



PGN & FEN Notation

PGN (Portable Game Notation) is a plain-text format used to record entire chess games, including moves, metadata (like player names and event), and annotations. It allows games to be easily shared, stored, or analyzed by humans and software.

Algebraic notation is the standard format used for recording the moves inside a PGN file.

FEN (Forsyth-Edwards Notation), on the other hand, describes a *single* position on the board using a compact string—useful for saving, resuming, or analyzing specific moments in a game rather than the full move history.

The **FEN** string has **6 space-separated fields**, in this order:
1.Piece placement
2.Active color
3.Castling availability
4.En passant target square
5.Halfmove clock: Number of half-moves since the last pawn move or capture, used to determine if a draw can be claimed under the fifty-move rule.
6.Fullmove number: Indicates white's move number.

PGN & FEN Notation

FEN Notation Decoding

1. Piece Placement (First field):
rn1qkbnr/ppp1pppp/4b3/3p4/4P3/2N5/PPP1PPPP/R1BQKBNR
This shows the board rank by rank from 8 to 1, separated by /. Each letter stands for a piece:
Lowercase = black and Uppercase = white pieces
Numbers = empty squares
Breaking it down:
8th rank: rn1qkbnr → rook, knight, one empty square, queen, king, bishop, knight, rook
7th rank: ppp1pppp → 3 pawns, 1 empty, 4 pawns
...
2. Active Color (Second field): w
White to move.
3. Castling Availability (Third field): *KQkq*
K: White can castle kingside
Q: White can castle queenside
k: Black can castle kingside
q: Black can castle queenside
4. En Passant Target Square (Fourth field): -
No en passant move is currently available.
5. Halfmove Clock (Fifth field): *2* **& 6. Fullmove Number (Sixth field): 3**

Computer Analysis of Chess

Chess Engines

A **chess engine** is a computer program that plays chess by calculating millions of possible move sequences and evaluating their outcomes to choose the best move. It's like a calculator for chess positions - except much smarter and faster than any human.

Stockfish, Leela Chess Zero (LCZero), and Komodo Dragon are three of the most popular and powerful chess engines today.

Stockfish binary can be downloaded from https://stockfishchess.org/download/

UCI
Universal Chess Interface (UCI) is an open communication protocol that enables chess engines to communicate with user interfaces.

Python-chess Library

- Can parse PGN and FEN of games
- Can interact with chess engines for analysis, using UCI standard
- Installable using `*pip install chess*`

Code for PGN Parsing and making the moves:

```
import chess.pgn

pgn = open("data/pgn/kasparov-deep-blue-1997.pgn")
game = chess.pgn.read_game(pgn)
game.headers["Event"] # 'IBM Man-Machine, New York USA'

# Iterate through all moves and play them on a board.
board = game.board()
for move in game.mainline_moves():
    board.push(move)

print(board) # Prints FEN representation
# Board('4r3/6P1/2p2P1k/1p6/pP2p1R1/P1B5/2P2K2/3r4 b - - 0 45')
```

Load a chess engine and make the engine play itself until the game ends. 100 ms time is given to make each move:

```
import chess
engine = chess.engine.SimpleEngine.popen_uci(
    "path_to\stockfish_14_win_x64_avx2.exe")

board = chess.Board()
while not board.is_game_over():
    result = chess.engine.play(board,
chess.engine.Limit(time=0.1))
    board.push(result.move)

engine.quit()
```

Analyzing moves using the engine:

Centipawn loss is a metric used to measure how much worse a move is compared to the best possible move, according to a chess engine - where 100 centipawns = 1 pawn.

A score of +0.80 means White is better by the equivalent of 0.8 pawns. A score like Cp(50) means the engine thinks the player to move is better by half a pawn (50 centipawns). If the score is Mate(3), it means there's a forced checkmate in 3 moves. A score of +1.00 (Cp(100) or 100 centipawns) could mean "White (or the next to move) is as good as being one pawn up"

For example, if the best move evaluates to +0.80 and the move played evaluates to +0.50, the centipawn loss is 30.

```
engine = chess.engine.SimpleEngine.popen_uci("/usr/bin/
stockfish")

board = chess.Board()
info = engine.analyse(board,
chess.engine.Limit(time=0.1))
print("Score:", info["score"])
# Score: PovScore(Cp(+20), WHITE)

board = chess.Board("r1bqkbnr/
p1pp1ppp/1pn5/4p3/2B1P3/5Q2/PPPP1PPP/RNB1K1NR w KQkq - 2
4")
info = chess.engine.analyse(board,
chess.engine.Limit(depth=20))
print("Score:", info["score"])
# Score: PovScore(Mate(+1), WHITE)
```

Chess Board Library

- Installable as `*pip install chess-board*`
- Uses *pygame* to visualize the chess board

```
from chessboard import display

valid_fen = 'rnbqkbnr/pp1ppppp/8/2p5/4P3/5N2/PPPP1PPP/
RNBQKB1R b KQkq - 1 2'

# Initialization - Will display the board UI on screen
game_board = display.start()

# Position change/update
display.update(valid_fen, game_board)
```

Building a chess analyzer

- Accept PGN
- Analyze with Engine (Stockfish)
- Provide move by move analysis and best move suggestion.
- Uses chess & chess-board libraries for analysis and visualization

Full code at: https://github.com/adarshdigievo/talks

Application screenshots: