My First Python Chess Bot

Sign-in here!

(and log into Github!)

Start Up Environment

(Thanks for coming!)

1. **Install Python 3!**
   a. https://www.python.org/downloads/

   OR:

   1. Login to Debian environment
   2. Download pip (https://shorturl.at/zKPW8)
   3. Go into the Download directory via cd ~/Download
   4. python get-pip.py
   5. take notes of the local bin directory pip installed to
   6. cd /home/<your_netid>/.local/bin

2. **Install required libraries!**
   a. In a terminal (with Python3 installed), enter the following commands:

   ./pip install chess

   ./pip install pygame

3. **Clone the workshop repository!**
   a. https://github.com/Marlo-Ong/ACM-ChessBotWorkshop-Python

The Base Code

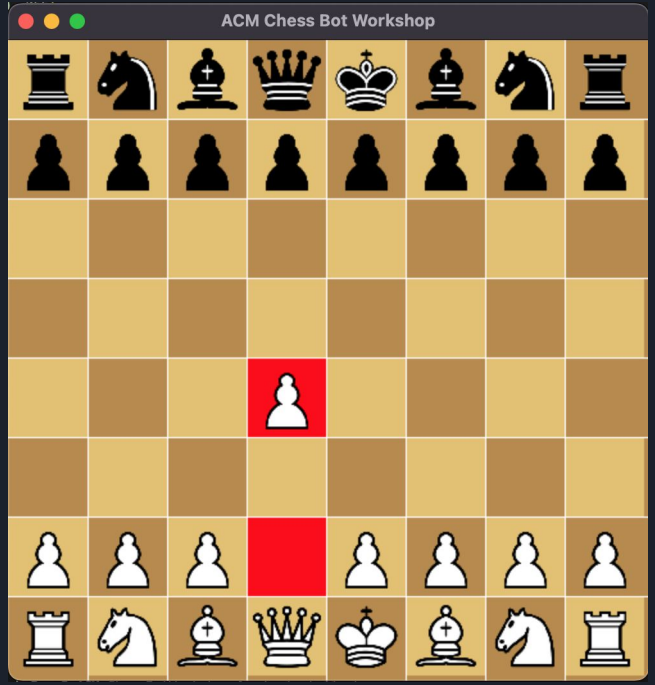
1. **Run the GUI:**

   ```
   python3 main.py
   ```

   Controls: “R” to restart, CTRL+Z to undo

2. **Check the code given to you!**
   a. main.py: the front-end; **no need to touch this** unless you’re adding new features!
   b. ChessBot.py: the brains of the engine. **You’ll be making changes here!**

3. **Currently, our bot is making random moves… how can we make it better?**



Cool PYTHON Syntax!!1

Ternary Operators:

```
# C++
   string WHAT = (comparison > 1) ? "done" : "gone"
# Python
   WHAT = "done" if (comparison > 1) else "gone"
```

Tuple Unpacking:

```
# Python
def gottem():
       return "nine", 10
       # = return (“nine”, 10)
v1, v2 = gottem()
```

Funny things…

```
# Unpacking with Underscore
   _, v2 = ("1", 2)

# Infinity
   float("inf")
```

Class Syntax:

```
class Classy:

   # Parameterized constructor
   def __init__(self, arg):
       self.arg = arg
       self.foo(); # self = this

   def foo(self):
       print(“Hello World!”)

# Actually structs! Default public
myObj = Classy(25)
print(myObj.arg)
```

Basics of a Chess Engine

How does it work? Search and Evaluation!

For each given board state and **max depth**:

1. **Search** all possible moves that can be made
2. **Evaluate** all moves for how in favor a player is
3. **Return** the move with the highest score
4. **Play** the returned move, then start all over again!
    a. (Loop x number of times equal to the **max depth)**

*Now let's start with the **search.***

What is Depth and WHY?
There are ~30 possible moves for each board, and ~40 moves each game.

So, the number of possible chess game variations (Shannon number) is:

$$30^{40} = 10^{120}!$$

Currently, a chess engine doesn't have the memory and speed to evaluate entire games up to this number!

So, a **depth** of an engine's search algorithm is the limit of how many moves it will search in advance.
(3-4 is typically good for the speed)

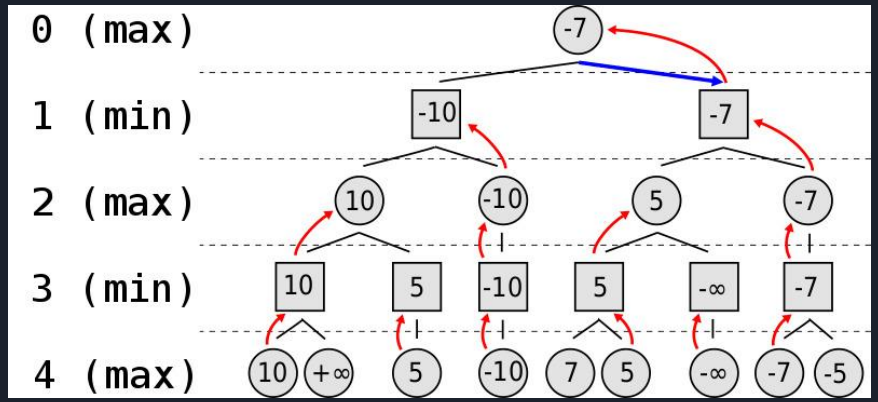
Making it Better! (Minimax Algorithm)

What is minimaxing?

- “The moves where the maximizing player wins are assigned with positive infinity, while the moves that lead to a win of the minimizing player are assigned with negative infinity.”

Changes:

- We start from depth = 1, not depth = 0. (Our bot plays on ODD moves)
- “maximizingPlayer” will be defined inside the minimax algorithm instead of being passed as an argument




```
function minimax(node, depth, maximizingPlayer) is
    if depth = 0 or node is a terminal node then
        return the heuristic value of node
    if maximizingPlayer then
        value := −∞
        for each child of node do
            value := max(value, minimax(child, depth − 1, FALSE))
        return value
    else (* minimizing player *)
        value := +∞
        for each child of node do
            value := min(value, minimax(child, depth − 1, TRUE))
        return value
```

Evaluation: Piece Values

Almost all chess engines evaluate boards based on **cumulative piece value** - a player with a queen is better than a player with none!

via https://en.wikipedia.org/wiki/Chess_piece_relative_value

Symbol					
Piece	pawn	knight	bishop	rook	queen
Value	1	3	3	5	9

What's the best way to do this? **Dictionaries!**

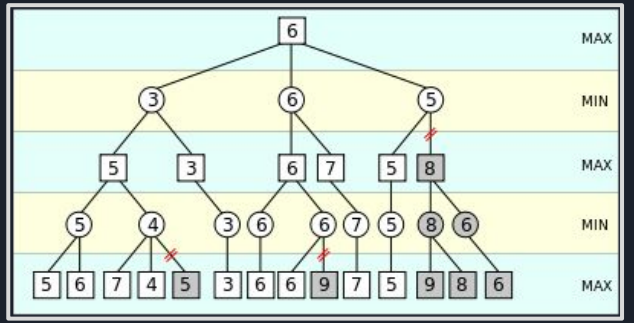
Making it Better! (Alpha-Beta Pruning)

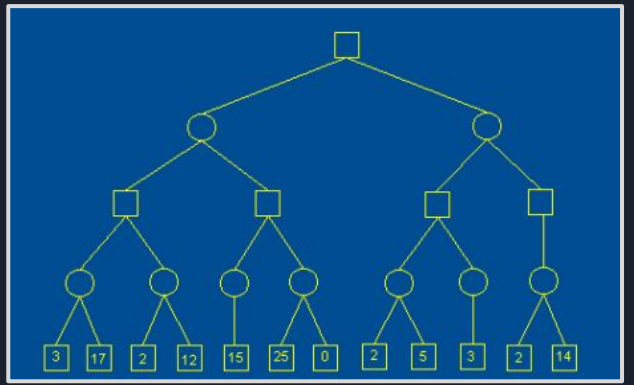
What is alpha-beta pruning?

- “Alpha–beta pruning is a search algorithm that seeks to decrease the number of nodes that are evaluated by the minimax algorithm in its search tree.”
- “It stops evaluating a move when at least one possibility has been found that proves the move to be worse than a previously examined move. Such moves need not be evaluated further.”

number of leaves with depth n and b = 40

depth n	b^n	$b^{\lceil n/2 \rceil} + b^{\lfloor n/2 \rfloor} - 1$
0	1	1
1	40	40
2	1,600	79
3	64,000	1,639
4	2,560,000	3,199
5	102,400,000	65,569
6	4,096,000,000	127,999
7	163,840,000,000	2,623,999
8	6,553,600,000,000	5,119,999





Credits

- python-chess Documentation: https://python-chess.readthedocs.io/en/latest/core.html
- Chess Pygame Frontend: https://blog.devgenius.io/simple-interactive-chess-gui-in-python-c6d6569f7b6c
- Wikipedia: Alpha-Beta Pruning Pseudocode: https://en.wikipedia.org/wiki/Alpha%E2%80%93beta_pruning#Pseudocode
- ChessProgramming.org: https://www.chessprogramming.org/Main_Page

THANK YOU!