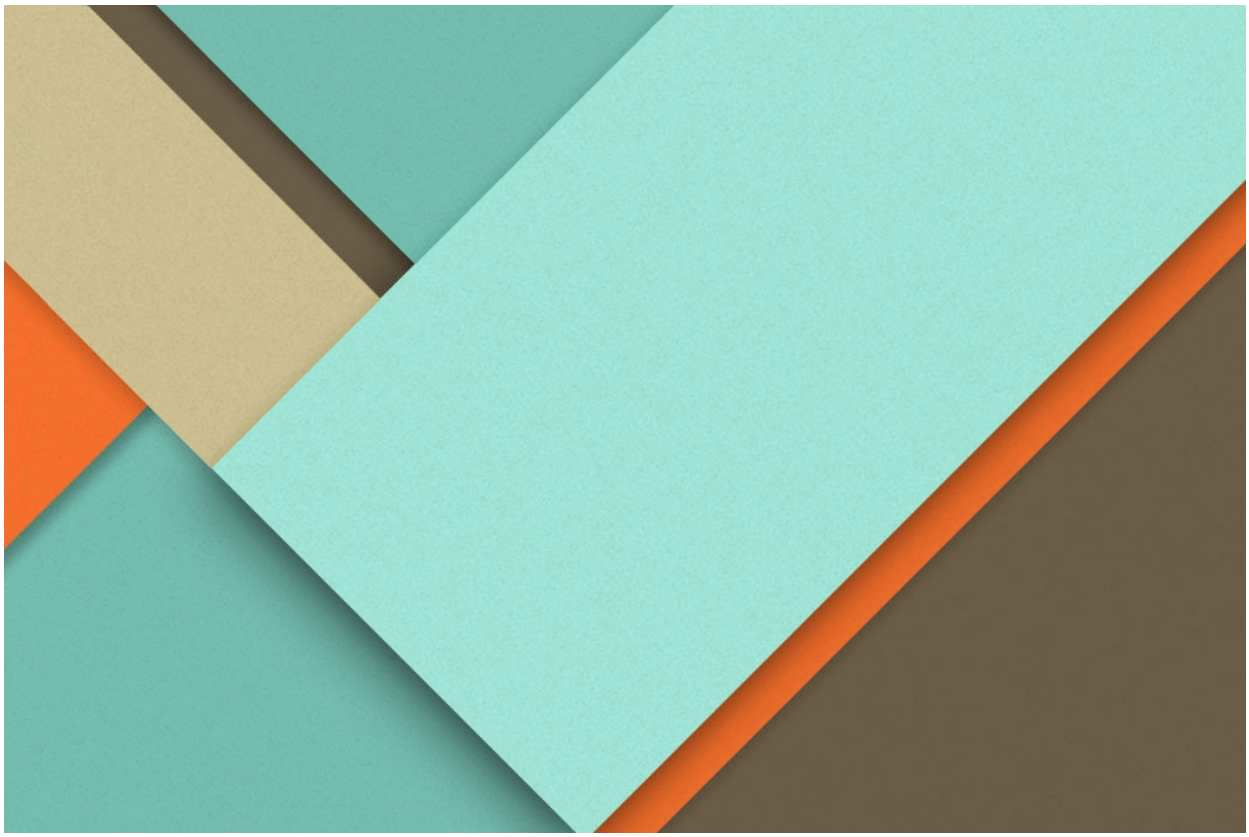
ARTIFICIAL INTELLIGENCE PROJECT



CHESS GAME !

Move analyzing

G Hemant Lakshman Roll no. 21

Rutwik Babu Roll no. 29

SUBMITTED TO: DR. SAGAR PANDE

Section : K18KK

School of Computer Science

LOVELY PROFESSIONAL UNIVERSITY , JALANDHAR

ABSTRACT:

This project is about a python chess game which use's pygame library to implement a 2V2 game and tkinter for GUI.Now in a 2V2 game the game shows the vulnerable moves in red color gradient and warns the user before the move which would help the user to learn chess in initial stages of their playtime and improves their skills in the game.

There is a second module in this game in which we can play with AI and it is for testing our skills in a tireless session of games played by AI.

The algorithm or principle we have used is minimax and alpha-beta pruning.

Some more concepts have been handy while making this project which will be elaborated in the upcoming sections.

RELATED WORK:

We have been looking a lot of articles from "MEDIUM" as well other online platforms to get help and ideas in completing our project

But the article which solved most of our query and was sensible for our needs was by Lauri Hartikka

Here are the points we learned from the article

- move-generation
- board evaluation

- minimax
- and alpha beta pruning.

It was difficult for us to interpret her ideas into code as the maximum of their concepts were based on javascript and a lot of features were included in the vast libraries provided by the language.

GUI part also wasn't easy to interpret as it was written in HTML and clearly she was trying to build a web application.But the reason behind picking her name as an influencer was clarity in concepts and the way she made it clear so everyone reading her article was able to implement her ideas.

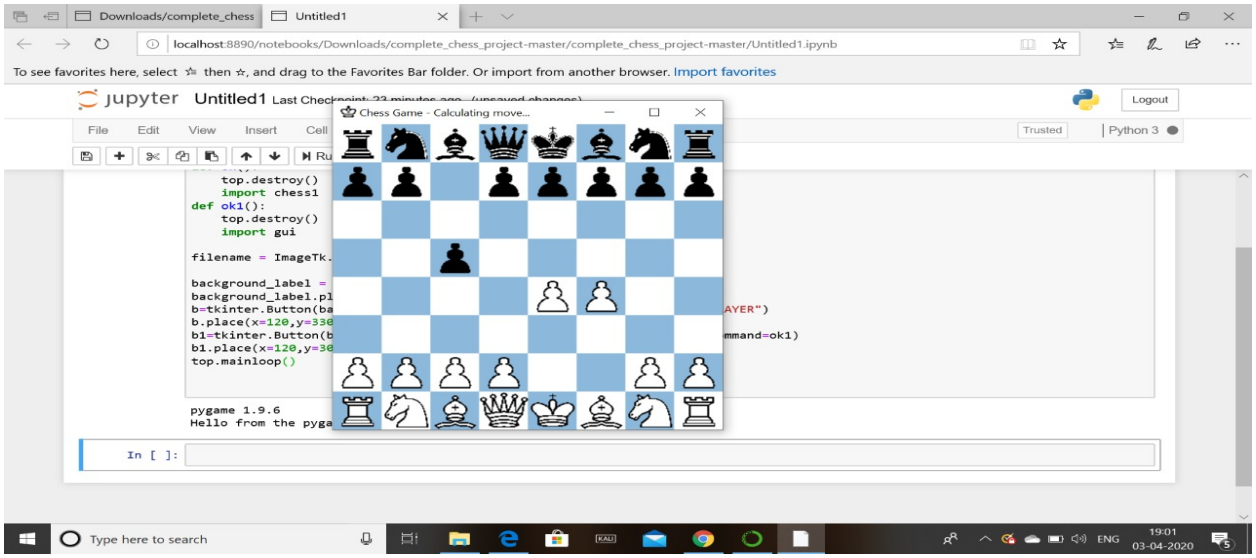
Lot of help was taken from "stack overflow" regarding errors and syntax while we were trying to implement the algorithm in python using "aigames".

IMPLEMENTATION:

1. **GUI:** For GUI we have used “Tkinter” library and adding to that used the “pygames.display” library for move generation, and for visualizing the board. The move generation library basically implements all the rules of chess. Based on this, we can calculate all legal moves for a given board state.

Using these libraries will help us focus only on the most interesting task: creating the algorithm that finds the best move.
We’ll start by creating a function that just returns a random move from all of the possible moves:

Although this algorithm isn’t a very solid chess player, it’s a good starting point, as we can actually play against it.



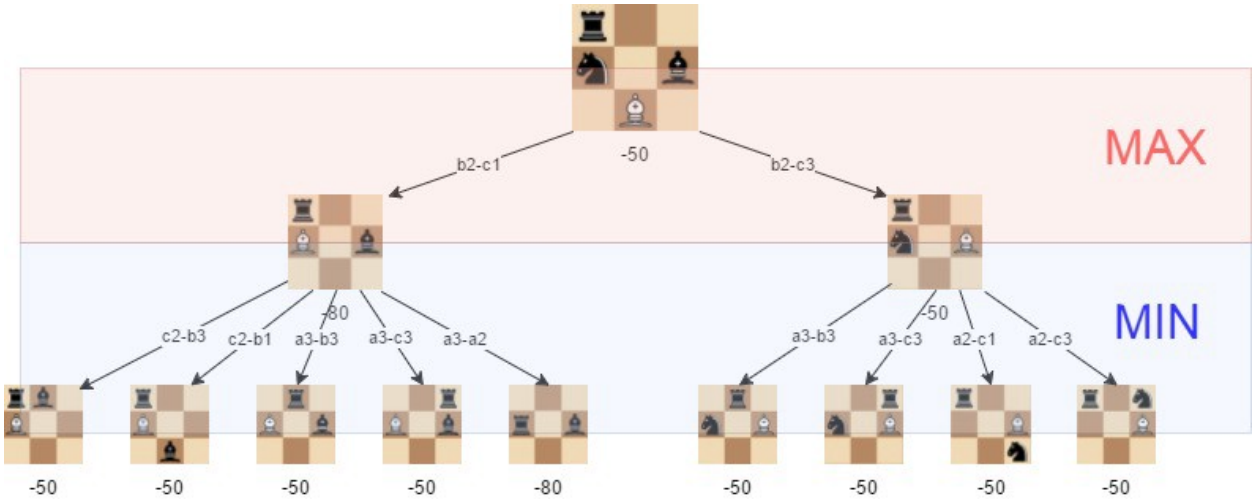
Now let's try to understand which side is stronger in a certain position. The simplest way to achieve this is to count the relative strength of the pieces on the board using the following table:

2.Search tree using Minimax:

Next we're going to create a search tree from which the algorithm can choose the best move. This is done by using the Minimax algorithm.

In this algorithm, the recursive tree of all possible moves is explored to a given depth, and the position is evaluated at the ending "leaves" of the tree.

After that, we return either the smallest or the largest value of the child to the parent node, depending on whether it's a white or black to move. (That is, we try to either minimize or maximize the outcome at each level.)



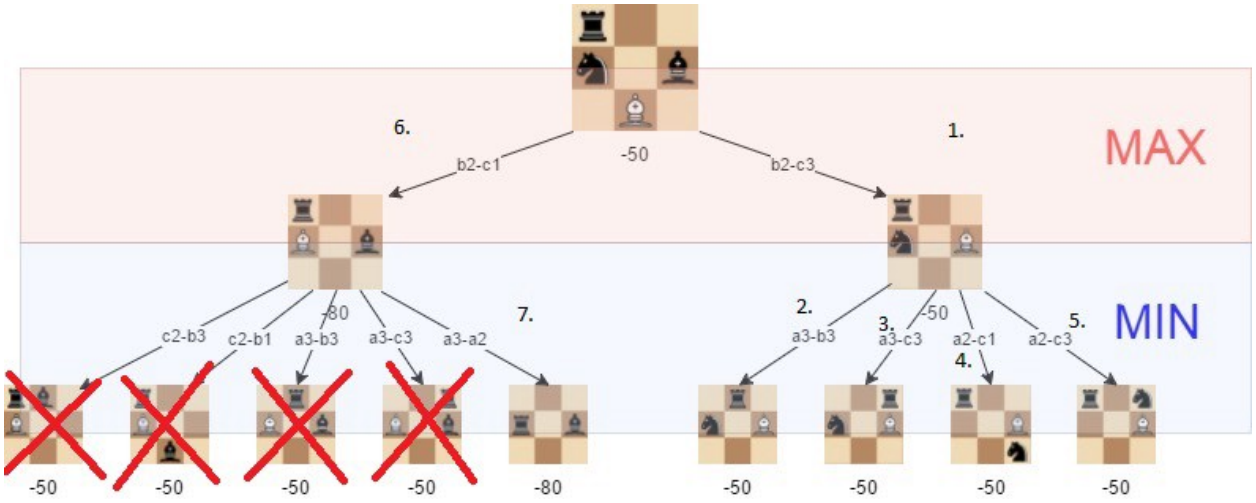
3.Alpha-beta pruning

Alpha-beta pruning is an optimization method to the minimax algorithm that allows us to disregard some branches in the search tree. This helps us evaluate the minimax search tree much deeper, while using the same resources.

The alpha-beta pruning is based on the situation where we can stop evaluating a part of the search tree if we find a move that leads to a worse situation than a previously discovered move.

The alpha-beta pruning does not influence the outcome of the minimax algorithm — it only makes it faster.

The alpha-beta algorithm also is more efficient if we happen to visit first those paths that lead to good moves.



4.Improved evaluation function:

The initial evaluation function is quite naive as we only count the material that is found on the board. To improve this, we add to the evaluation a factor that takes in account the position of the pieces. For example, a knight on the center of the board is better (because it has

more options and is thus more active) than a knight on the edge of the board.

We'll use a slightly adjusted version of the piece-square table.

```
[ -3.0, -4.0, -4.0, -5.0, -5.0, -4.0, -4.0, -3.0],
[ -3.0, -4.0, -4.0, -5.0, -5.0, -4.0, -4.0, -3.0],
[ -3.0, -4.0, -4.0, -5.0, -5.0, -4.0, -4.0, -3.0],
[ -3.0, -4.0, -4.0, -5.0, -5.0, -4.0, -4.0, -3.0],
[ -2.0, -3.0, -3.0, -4.0, -4.0, -3.0, -3.0, -2.0],
[ -1.0, -2.0, -2.0, -2.0, -2.0, -2.0, -2.0, -1.0],
[  2.0,  2.0,  0.0,  0.0,  0.0,  0.0,  2.0,  2.0 ],
[  2.0,  3.0,  1.0,  0.0,  0.0,  1.0,  3.0,  2.0 ]
```

```
[ -2.0, -1.0, -1.0, -0.5, -0.5, -1.0, -1.0, -2.0],
[ -1.0,  0.0,  0.0,  0.0,  0.0,  0.0,  0.0, -1.0],
[ -1.0,  0.0,  0.5,  0.5,  0.5,  0.5,  0.0, -1.0],
[ -0.5,  0.0,  0.5,  0.5,  0.5,  0.5,  0.0, -0.5],
[  0.0,  0.0,  0.5,  0.5,  0.5,  0.5,  0.0, -0.5],
[ -1.0,  0.5,  0.5,  0.5,  0.5,  0.5,  0.0, -1.0],
[ -1.0,  0.0,  0.5,  0.0,  0.0,  0.0,  0.0, -1.0],
[ -2.0, -1.0, -1.0, -0.5, -0.5, -1.0, -1.0, -2.0]
```

```
[  0.0,  0.0,  0.0,  0.0,  0.0,  0.0,  0.0,  0.0],
[  0.5,  1.0,  1.0,  1.0,  1.0,  1.0,  1.0,  0.5],
[ -0.5,  0.0,  0.0,  0.0,  0.0,  0.0,  0.0, -0.5],
[ -0.5,  0.0,  0.0,  0.0,  0.0,  0.0,  0.0, -0.5],
[ -0.5,  0.0,  0.0,  0.0,  0.0,  0.0,  0.0, -0.5],
[ -0.5,  0.0,  0.0,  0.0,  0.0,  0.0,  0.0, -0.5],
[ -0.5,  0.0,  0.0,  0.0,  0.0,  0.0,  0.0, -0.5],
[  0.0,   0.0, 0.0,  0.5,  0.5,  0.0,  0.0,  0.0]
```

```
[ -2.0, -1.0, -1.0, -1.0, -1.0, -1.0, -1.0, -2.0],
[ -1.0,  0.0,  0.0,  0.0,  0.0,  0.0,  0.0, -1.0],
[ -1.0,  0.0,  0.5,  1.0,  1.0,  0.5,  0.0, -1.0],
[ -1.0,  0.5,  0.5,  1.0,  1.0,  0.5,  0.5, -1.0],
[ -1.0,  0.0,  1.0,  1.0,  1.0,  1.0,  0.0, -1.0],
[ -1.0,  1.0,  1.0,  1.0,  1.0,  1.0,  1.0, -1.0],
[ -1.0,  0.5,  0.0,  0.0,  0.0,  0.0,  0.5, -1.0],
[ -2.0, -1.0, -1.0, -1.0, -1.0, -1.0, -1.0, -2.0]
```

```
[-5.0, -4.0, -3.0, -3.0, -3.0, -3.0, -4.0, -5.0],
[-4.0, -2.0,  0.0,  0.0,  0.0,  0.0, -2.0, -4.0],
[-3.0,  0.0,  1.0,  1.5,  1.5,  1.0,  0.0, -3.0],
[-3.0,  0.5,  1.5,  2.0,  2.0,  1.5,  0.5, -3.0],
[-3.0,  0.0,  1.5,  2.0,  2.0,  1.5,  0.0, -3.0],
[-3.0,  0.5,  1.0,  1.5,  1.5,  1.0,  0.5, -3.0],
[-4.0, -2.0,  0.0,  0.5,  0.5,  0.0, -2.0, -4.0],
[-5.0, -4.0, -3.0, -3.0, -3.0, -3.0, -4.0, -5.0]
```

```
[0.0,  0.0,  0.0,  0.0,  0.0,  0.0,  0.0,  0.0],
[5.0,  5.0,  5.0,  5.0,  5.0,  5.0,  5.0,  5.0],
[1.0,  1.0,  2.0,  3.0,  3.0,  2.0,  1.0,  1.0],
[0.5,  0.5,  1.0,  2.5,  2.5,  1.0,  0.5,  0.5],
[0.0,  0.0,  0.0,  2.0,  2.0,  0.0,  0.0,  0.0],
[0.5, -0.5, -1.0,  0.0,  0.0, -1.0, -0.5,  0.5],
[0.5,  1.0, 1.0,  -2.0, -2.0,  1.0,  1.0,  0.5],
[0.0,  0.0,  0.0,  0.0,  0.0,  0.0,  0.0,  0.0]
```

-> 2V2 PART:

1.The 2V2 part was easy compared to the AI part we used pygames and its function to get the game working.

2.We had created classes for every type of piece present in the chess board which had set of allowed moves or legal moves which are displayed as green in color in chess board as we click on the piece we want to play with.

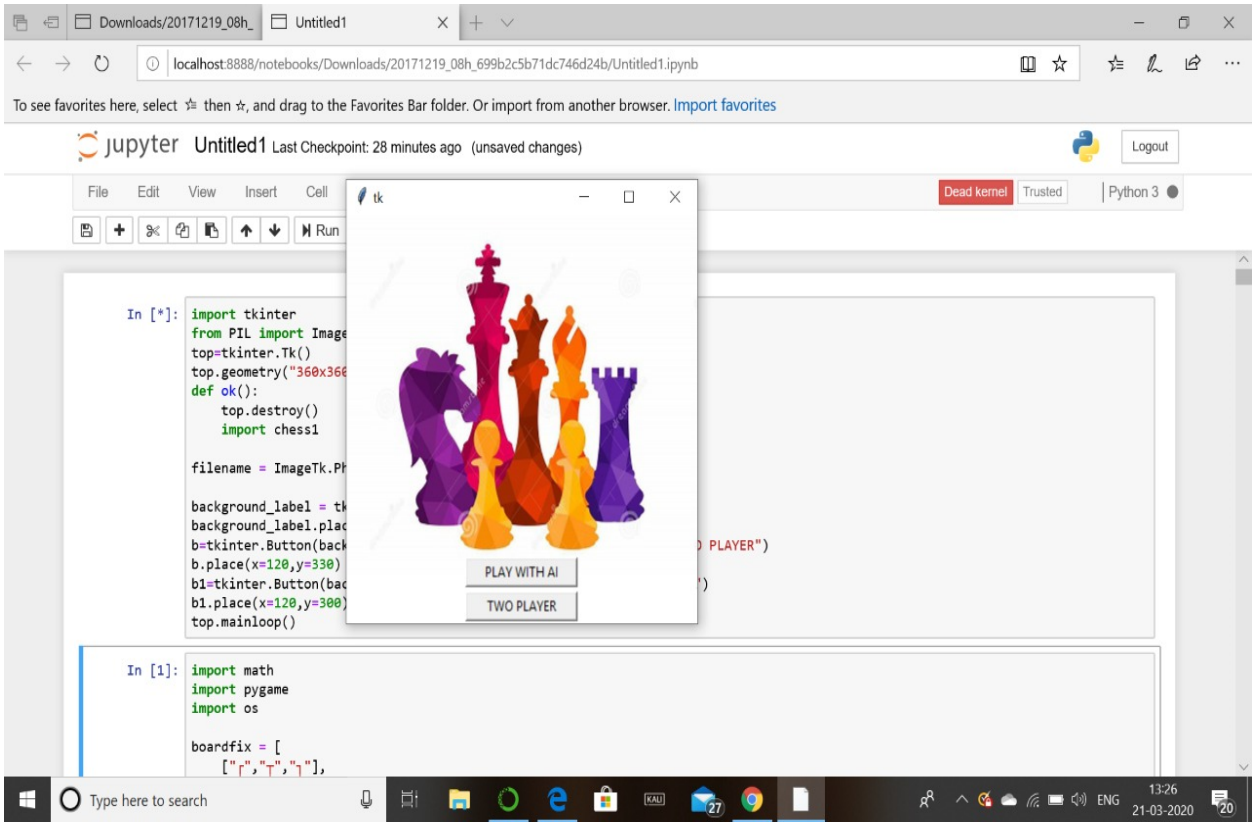
3.We had used a multidimensional array which corresponds the positions in the chess boards and the changes are done in these arrays which are further raycasted on to the chessboard.

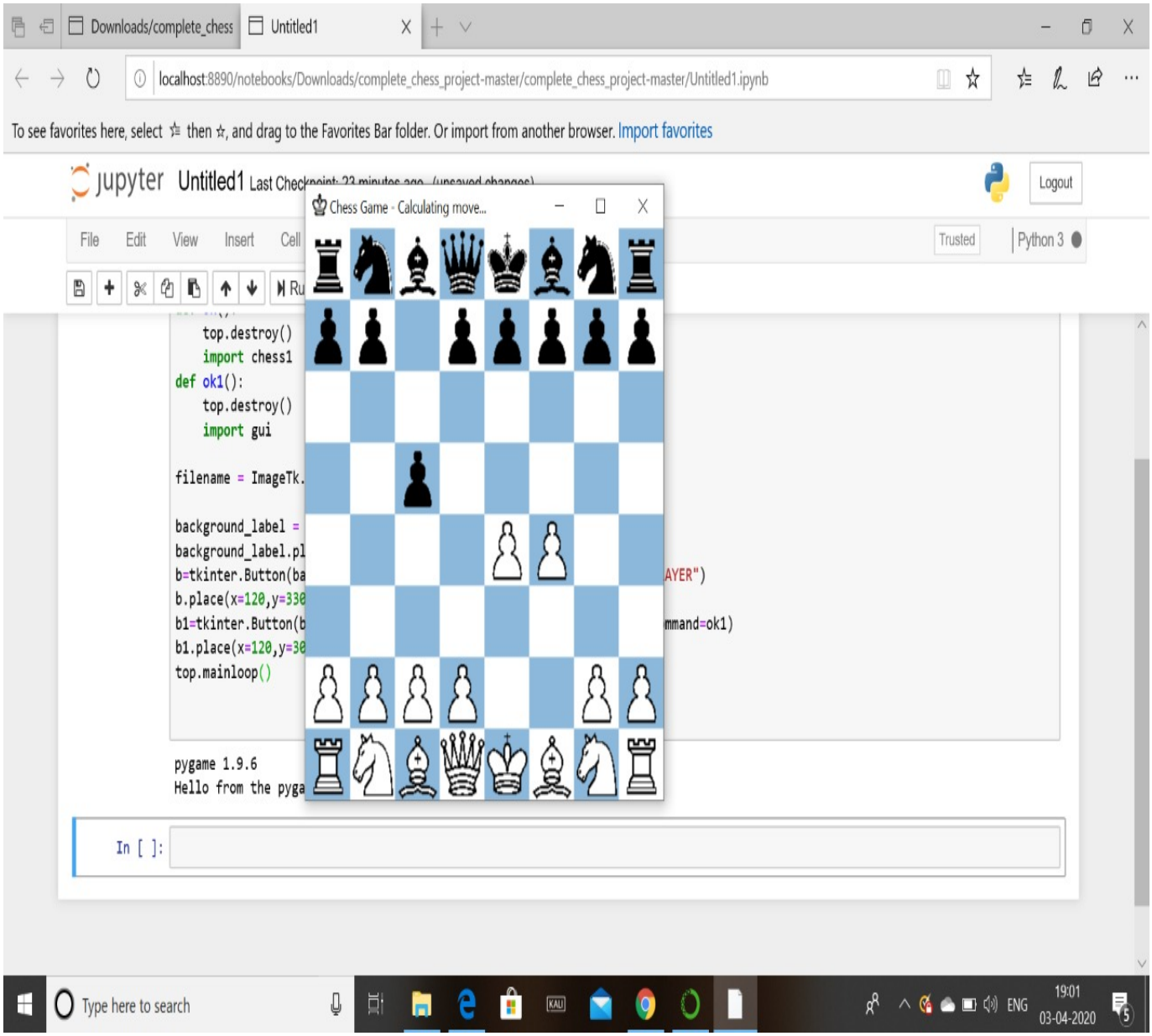
4.Raycast is simply the changes we do in the array lead to corresponding changes in the chessboard.

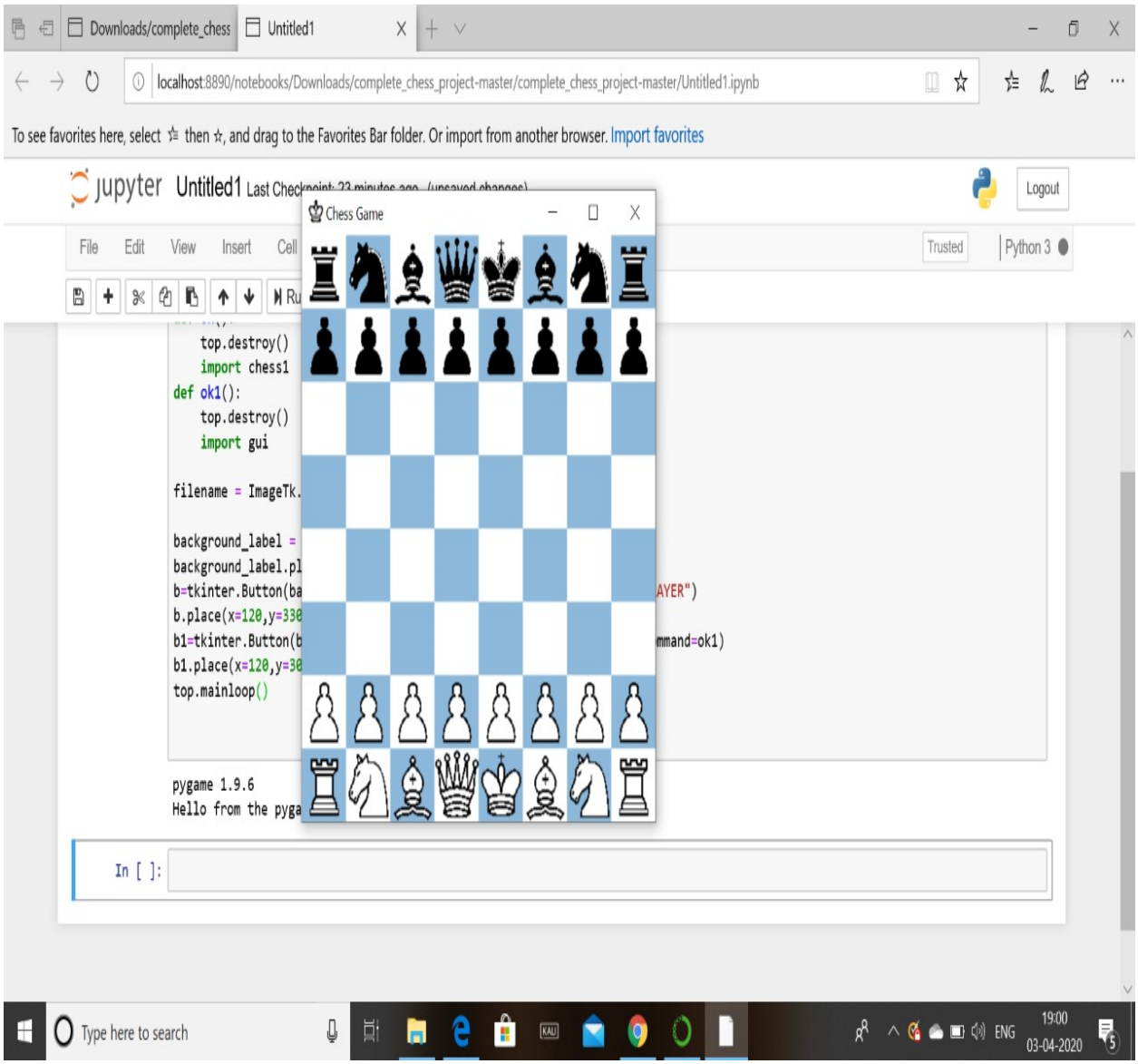
5.And the move review system is included in this module of the game which shows red color in the board which ahs threat of being eliminated if we do the move.

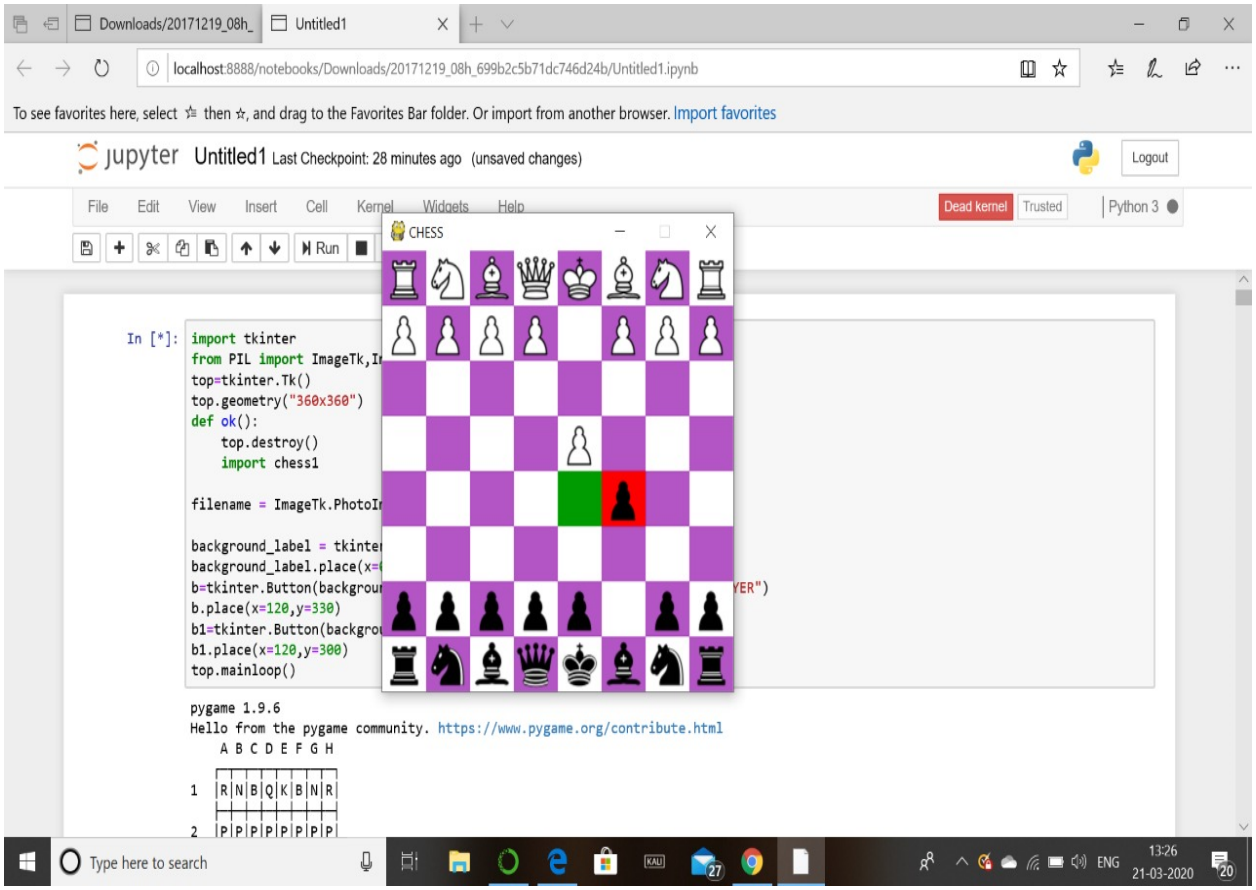
6.A lot of handy functions are used included in the pygames library.

SCREENSHOT' s/RESULT:




Downloads/complete_chess
Untitled1
localhost:8890/notebooks/Downloads/complete_chess_project-master/complete_chess_project-master/Untitled1.ipynb
To see favorites here, select then , and drag to the Favorites Bar folder. Or import from another browser. Import favorites
jupyter Untitled1
Logout
File Edit View Insert Cell
Trusted
Python 3
Chess Game - Calculating move...
top.destroy()
import chess1
def ok1():
top.destroy()
import gui
filename = ImageTk.
background_label =
background_label.pl
b=tkinter.Button(ba
b.place(x=120,y=330
b1=tkinter.Button(b
b1.place(x=120,y=30
top.mainloop()
AYER")
mmand=ok1)
pygame 1.9.6
Hello from the pyga
In [ ]:
Type here to search
ENG
19:01
03-04-2020


Downloads/complete_chess
Untitled1
localhost:8890/notebooks/Downloads/complete_chess_project-master/complete_chess_project-master/Untitled1.ipynb
To see favorites here, select then , and drag to the Favorites Bar folder. Or import from another browser. Import favorites
jupyter Untitled1
Logout
File Edit View Insert Cell
Trusted
Python 3
Chess Game
top.destroy()
import chess1
def ok1():
top.destroy()
import gui
filename = ImageTk.
background_label =
background_label.pl
b=tkinter.Button(ba
b.place(x=120,y=330
b1=tkinter.Button(b
b1.place(x=120,y=30
top.mainloop()
AYER")
mmand=ok1)
pygame 1.9.6
Hello from the pyga
In [ ]:
Type here to search
ENG
19:00
03-04-2020



IMPORTANT LIBRARIES USED:

1.Pygame - import pygame,chess

2.Tkinter

3.random -from random import choice

4.traceback - from traceback import format_exc

5.sys - from sys import stderr

6.time -from time import strftime

7.copy - from copy import deepcopy

TEAM RESPONSIBILITIES:

1.Repots section was divided as we have to submit two reports

Initial reports was written by RUTWIK BABU

Final report is written by G HEMANT LAKSHMAN

2.The code section was equally done by both of us. We first planned all the components and shared the pseudo codes with each other.

Intelligence was used on both the ends at each and every step only the typing part was the matter to be shared .

GUI - G HEMANT LAKSHMAN

2 versus 2 - RUTWIK BABU

AI - half code - G HEMANT LAKSHMAN

AI -other half code - RUTWIK BABU

REFERENCES:

1.We had some of our ideas from articles from medium and online articles.

https://www.freecodecamp.org/news/simple-chess-ai-step-by-step-1d55a9266977/

2.We have also checked on some repositories from github to get an idea of implementation.

3.And a lot of help from stack overflow to get ideas related errors and syntax