UPDATE FOR TP3:
I finished the minimax AI and it now plays for the opponent if you select the player vs AI mode. I'm going to focus on bettering the AI but if I have gotten a really good one, I'll start implementing opening positions. The game should be fully functioning now!

UPDATE FOR TP2:

I have finished everything except the minimax AI and opening position. I have implemented a start screen and you can navigate back and forth and choose the type of game you want to play. There were many many difficult bugs that I encountered and castling/en passant were extremely difficult to conceptualize and write for all the cases in which they are/aren't supposed to be legal. I have 3 python files, one that is the full game, one that is just to play as white, and one to just play as black. I import the one that is just to play as white, and one that is just to play as black into the full game and simply call their functions when needed.

I have started minimax but disregard for tp2 because it is very incomplete

Use playChess.py to play what I have done for mvp

What I plan to do: CHESS

STRUCTURAL PLAN:

I plan to use standard MVC animation programming. I will split up my work into two sections, one that initializes my app/draws on my canvas, and the other will do my computations (legal moves, AI, etc.) This will hopefully be sufficient in keeping my code organized.

I will also be commenting heavily so that I explain every problem that a person who just starts looking at the code will immediately understand why I do what I do

ex.) I have to make sure a king doesn't move into a check so I will explain that I have to do that in a specific function and explain my solution to the problem

I also will use some images so that I gain exposure to that part of tkinter and so that my chess looks niceee

ALGORITHMIC PLAN:

Hardest Parts:
- Legal Moves
- Castling
- En-Passant
- Minimax AI
- Finding what the opening position is

Legal Moves:

Everything except the pawn and king will be easy, all I have to do
is implement a word-search strategy to find all possible, legal moves

The king gets tricky because it has some special features like being checked/checkmated
and not being able to walk into the line of fire

For the king I will try to keep the logic as clean as possible by having
a parameter saying whether or not we should check the opponents legal moves
so if we are already checking an opponent's legal moves, we don't do it again
Also we should check if the king is currently in check. If it is, then we move
the king and check if it is still in check. If it is then we move it back to its current location

I haven't figured out pawns yet

Castling/En-Passant:

Implementing castling will be a matter of keeping a list of moves made
and hard-coding pressing a king and trying to move it to a specific square.
If the king or rook are in the list of moves made then you can't castle

I haven't figured out En-Passant yet

Minimax AI:

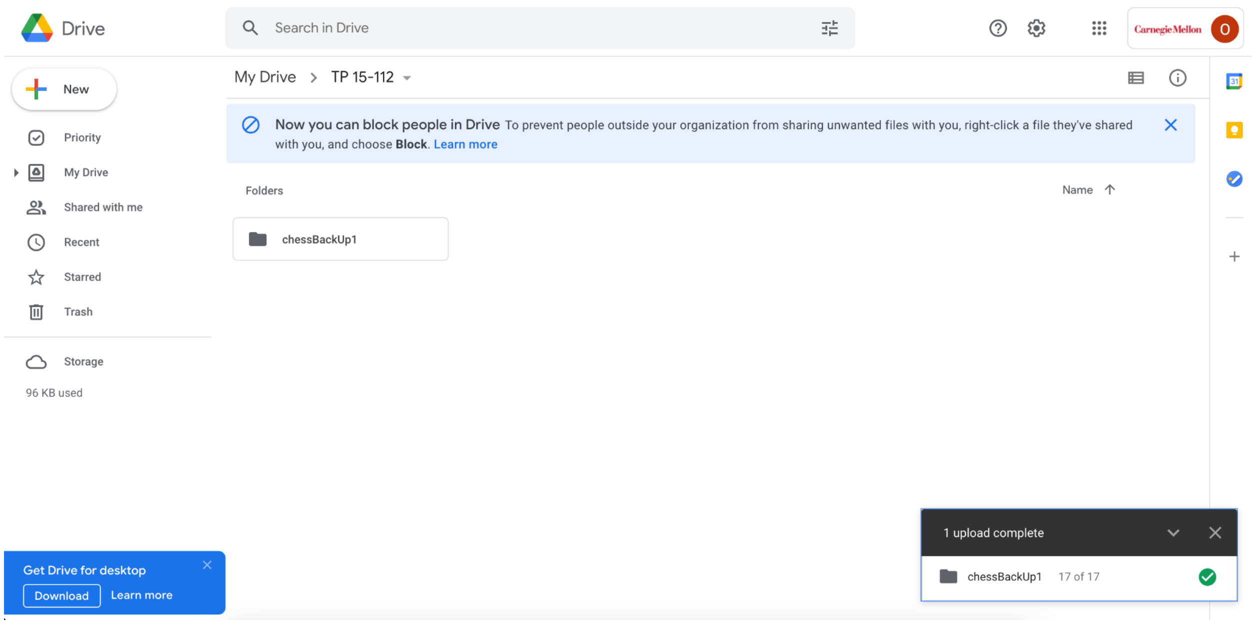
After reading https://en.wikipedia.org/wiki/Minimax to find out
what minimax is, I have a sense that I need to recursively check all possible moves
until I reach a desired depth. Once I've reached that depth I can start checking
my evaluations to one another

Opening Positions:

I really want to scour a website for opening positions but for now
it seems like I will have to have a text file with a bunch of positions
and if the moveset is that position then write it

VERSION CONTROL PLAN:

I'm going to use google docs to store my back-ups


Drive

Search in Drive

Carnegie Mellon

My Drive > TP 15-112

New
Priority
My Drive
Shared with me
Recent
Starred
Trash
Storage
96 KB used

Now you can block people in Drive To prevent people outside your organization from sharing unwanted files with you, right-click a file they've shared with you, and choose **Block**. Learn more

Folders | Name

chessBackUp1

Get Drive for desktop
Download | Learn more

1 upload complete
chessBackUp1 17 of 17