Pimaster Chess Presentation outline

* (andrew) Intro: Howdy y’all. This is lindsey and cody, and I’m Andrew, and today we are going to show you our program, Pimaster Chess! (stats)
* (cody)Timer settings/menu/music: First off, we’ve got the menu. The play button will start the game, and the quit button will close the program. You have three possible settings for the timer: 15 minutes, 5 minutes, or no timer at all for more casual play. As you can see when we click play, the music will change dynamically.
* (andrew) The board is setup instantiating buttons iteratively to store in a dictionary with their position as keys. The pieces themselves are instances of subclasses of a piece superclass and stored in separate lists for each player. When a button is clicked, the program will check various factors to determine how to proceed, not allowing the user to input anything invalid and showing information on the side panel to let them know what action they can or cannot perform. When an initial piece corresponding to the color of the current player’s turn is selected, all of the valid moves of that piece will be calculated using the function from the corresponding piece subclass. If a piece is already selected and another tile is selected, the initially selected piece will move if the blank tile or piece to overtake is valid, deselect the piece if the same tile is clicked, switch over to the movement logic of another piece if another piece of the player’s color is selected, or give a message saying the move is invalid and allowing the player to input again. To allow for an easy learning experience for newer players, a highlight option is available to show the piece selected in green, possible blank tiles for movement in yellow, and possible pieces to overtake in red. The checkbox can be clicked at any point during gameplay to turn on and off highlights dynamically. The biggest piece of logic is taking into account the king. If the player is in check or would end up in check as a result of a move, that move will not be allowed, as you can see right now where certain pieces can’t move or are limited in movement due to being in check.
* (lindsey)Pawn swap/GPIO:
  + GPIO
    - Each piece type has a value, pawn at 1; rook, bishop, and knight at 2; queen at 3
    - Using these values, a score is determined for each player based on the pieces in play.
    - Using this score, a percentage is calculated and is used to divide the led line into the appropriate colors to display which player has the strategic advantage.
      * Each LED is programmatically responsible for a range of percentages. If the percentages of the scores fall in the range of an LED, that LED will be purple and all LEDs previously in the line will be red and all LEDs after will be blue.
    - The score of each player is only changed on two occasions: discarding a piece and swapping a pawn for a piece of higher value
    - When the pawn takes the bishop, the discard counter for the second player is updated, and the new scores are evaluated. The new scores have not changed enough for the lights to change…
    - This move gave the ability for the current player to swap the pawn for a piece of a higher strategic value. To this the pawn position and color is considered, and a new instance of a piece is created by the player’s choice using the side panel discard buttons. The pawn is then added to the discard and replaced with the new piece.
    - However, when the pawn is swapped for a new queen, the scores are reevaluated, and the new percent is high enough to alter the lighting of the LEDs.
* (andrew)checkmate: There are two possible win conditions setup, checkmate and, if the timer is on, running out of time. Every time a piece is moved, be it through moving to a blank tile, overtaking, pawn swapping, or a combination of these, a check is made to see if there is at least one possible move any piece on the opponent’s team could make. If there is one, the program will break from the logic and allow the player to move. If not, checkmate is reached and the game is over.
* (cody) popup window: As you can see, after a player has performed a checkmate (or the timer runs out), a victory message will appear, accompanied by music. [close the window and go back to the chess board] The tiles on the chess board do nothing, and this is the final state the game is left in. To exit the game, you can click the Quit button in the bottom-right corner.
* (lindsey) lessons learned\future improvements:
  + Lessons learned:
    - Improved python coding skills such as the use of Tkinter for the GUI and the implementation of GPIO.
    - The importance of good programming practices such as low coupling to ensure changes to the program would not break other parts of the program. High cohesion allowed for functions to be used in various places in the program easily without having to modify the functions to fit overly specific needs.
    - Finally, we learned how important comments and communication are vital for a group coding project to go smoothly.
  + Future Improvements:
    - Improvements could be made in the future.
    - Some simple features could include animations for various piece actions and a button to view more descriptive chess rules in a pop-up window that can be viewed mid game.
    - Another addition could be to implement separate game modes for different scenarios, such as if the player wants to practice in preset chessboards for a different type of challenge.
    - Other unique chess moves such as castling and en passant could be implemented as well.
    - In terms of GPIO, 7 segment LEDs and a button could be added to emulate competitive chess.
    - Lastly, an option to compete against an AI could be implemented with various difficulty.
* (cody)conclusion: Overall, our final edition of the program, after over 200 commits across all 6 branches, contained 9 separate classes with 58 different functions, along with a hell of a lot of variables. So! Thank you guys so much for listening, and we hope you enjoyed it.