General Overview:

- Make sure to install pygame
    - pip3 install pygame
- Also install matplotlib
    - Pip3 install matplotlib
- To run:
    - python3 battleship.py
        - Must be in correct directory

Pygame:

- This game is built using pygame
- You can get started at this site (https://www.pygame.org/wiki/GettingStarted)
- Tutorials can also be found on that page
- Pygame has rectangle objects, which we use for the grids
    - Each grid is a 2-d array filled with rectangle objects
    - Placed Ships is also a 2-d array but it has an inner array for each ship
        - Each inner ship is filled with rectangle objects
- Helpful sites that we used:
    - https://stackoverflow.com/questions/33963361/how-to-make-a-grid-in-pygame
    - https://stackoverflow.com/questions/7415109/creating-a-rect-grid-in-pygame
    - https://stackoverflow.com/questions/10467863/how-to-remove-replace-text-in-pygame

battleship.py:

- This is the main file. It is the file that you run to run the game.
- The function that gets called to run the battleship is main(). This is called at the bottom in a way that prevents other files from running it when they import the file.
- main()
    - This function starts the pygame window and the screen object.
    - It initializes a 2-d array to store player 1's and player 2's ship board, and target board.
    - Each player has an array that stores their hits and misses.
    - Each player has a 2-D array storing their placed ships.
    - Each player also has a copy of their placed ships. This is the array that a rectangle gets removed from if the opponent hits it.
- Broad Overview of logic:
    - While the game hasn't ended a while loop runs
    - If player 1 hasn't placed their ships, it passes the logic to place_ships.py
    - If player 2 hasn't placed their ships, it passes the logic to place_ships.py
    - If both players have placed their ships, it uses a boolean player1turn to track whose turn it is
    - It reacts to clicks and makes the user click until they have clicked a spot on the target board that they have not clicked yet
    - Then it checks if it is a hit or miss and updates the screen accordingly
    - After a move, there is a three second buffer
    - This loop runs until the game is over

    - It notifies via text if a ship is sunk
- A more in depth technical view can be found in the code files

add_text.py

- This file handles any and all text adding after the user has selected how many ships have been placed
- add_text(screen, text)
    - Prints the specified text at the top of the screen
- time_out(screen)
    - Prints that the game will be closing soon
    - This happens if their is more than 15 seconds in between ship part placements
- add_labels_ships(screen)
    - Adds column labels to ship board
- add_labels_target(screen)
    - Adds column labels to target board
- add_labels_middle(screen)
    - Adds row labels to the middle of the screen

place_ships.py

- This file handles the placement of player 1's and player 2's ships
- placePlayer1Ships(screen, ships, placedShips, shipBoard)
    - Uses ships to know what lengths of ships are needed
    - While the array of ships needed isn't empty, it will iterate through and get them placed
    - Similarly to battleship.py, it validates clicks and adds the ship to placedShips array for the respective player
    - It goes through each length in the ships array and makes sure the right number and length of ships are placed
- placePlayer2Ships is same as above but for player 2
- addShip()
    - Checks that ship is in a rectangle touching preexisting rectangles in that ship (using touchesShip())
    - Also checks that it is not part of another ship (using inShips())
- addToShips()
    - Actually adds rectangle to the ship
- More specific technical implementations are in code files

get_ships_num.py

- Asks user to select how many ships they would like for the game
- Creates rectangle for each choice and puts the number of corresponding ships in rectangle object
- Gets user's click to determine what number they selected
- From this, it sets the numbers in player1 and player2 ships and fills in the 2d array with the right number of empty arrays for the ship placements to go into

- Code specific and technical details are within the code files