1. ACCURACY AND ADHERENCE TO SPECIFICATION

I was successfully able to play your game and it followed the rule of peg solitaire - well done!

Your code successfully passed all illegal move tests! Well done.

The tkinter interface was a nice idea to try for a more advanced version of the game. For the most part this worked well. Please see my feedback on usability.

A log file was generated automatically and I was able to load a file. However, please see my feedback on and code execution RE load/save.

2. USABILITY OF GAME

Tkinter was a good choice to make a game. I was a little surprised that you chose not to use the buttons as a way of selecting and moving pegs. This would be a simple click event to capture the pegs. Nevertheless the the hybrid text entry tkinter game did work for you. It might be my eyesight, but I found the pegs quite hard to see. It was the choice of icons and very small size of the interface.

I thought the error messages when I was testing invalid moves were clear and informative - this is often overlooked in code! Well done.

The user guide needed improvement and didn't exist in any concrete form. The Jupyter notebook you provided was reasonably good (see next section), but this was more documentation that a user guide.

3. CODING STANDARDS

I recommend providing a notebook with a clear a title and a brief section describing how it is structured.

Overall I thought how you structured the notebook was good. For the most part included a brief overview of the functionality in markdown.

Your functions should still follow PEP8 guidelines and include a docstring describing goals, assumptions, parameters, return values etc. I recommend revising PEP8 and taking a look at some of the quick wins I recommend for the course https://www.pythonhealthdatascience.com/content/001_setup/prereq/05_pep8.html.

Overall the code was well organised in appropriately named functions - well done.

In a few cases you broke PEP8 guidelines for line length. Your code definitely needed linting. The link I've provided gives some advice on using flake8 in Juypter notebooks (see end).

You have used global variables. That's fine and doesn't affect your marks. General advice you will find online and in textbooks is not to use them as they potentially cause bugs - I think has caused you a v.minor one here with loading files. I tend to avoid them myself. Sometimes this can make your code a bit more complicated. You can take an OOP approach if needed and make use of class attributes (that have scope only to the class) or research/create what is called a singleton class. Its a little bit more tricky with functions as you are required to pass the variables as parameters and this can add up! The simplest advice I can give you is that if you are working with Jupyter notebooks you might want to declare "notebook level" variables near the start of the notebook as you did with dot_info. This at least makes the variables more visible to readers - at the moment several are declared inside functions which is confusing to read.

4. CODE EXECUTION

You needed to provide some form of details on the version of python that you used to develop the game e.g. was it 3.8, 3.9 or 3.10? or something else? I executed the code in the hds_code environment provided by the course (with no problems). So one option was to simply state you used the course provided environment.

There is a minor unhandled exception when the undo button is clicked before a move is taken. One simple way to prevent this exception is to disable the undo button until a move has been made (check the length of the moves list).

There's a minor bug with reading in solution files. The code works if you load the game and immediately load a solution file. If an attempt is made to load another file or load a file after making moves, the board does not reset and a user is presented with a (potentially large) number of invalid input messages. I attempted to reset by exiting the tkinter interface and re-executing the last cell in the notebook again. However, this results in the same errors. This can be solved by reseting the jupyter kernel and executing all cells again. This resets the board. I think the source may be the notebook level variable that represents the board.

I assume your code was tested. It would have been good to seen some evidence of testing of your code.