Question to consider:

* What sort of data object will we store the sudoku as for solving?
* How best to construct a solver? - Needed for checking that generated sudokus are solvable, for providing hints and for advanced difficulty classification beyond just the number of starting numbers given.

Main source: <https://github.com/norvig/pytudes/blob/main/ipynb/Sudoku%20IPython%20Notebook.ipynb>

* Data object used is a dictionary, allows us to consider the grid in terms of coordinates, ‘A1’, … , ‘B4’, … ,’I9’ for each square in the grid
  + Personal note: For our use case consider using a class object which stores the dictionary form of the sudoku puzzle along with other important information.
* Easiest way to solve: Back-propergation, by grouping squares into their row, column and block we can figure out what numbers it could possibly be by process of elimation, filling in wherever there is only one and repeating the process
* More difficult cases cannot just be solved by backpropagation, require more advanced methods which are often much more computation heavy
  + I.e. Alternative path methods where you pick a possible option for a square, assign it, then keep solving until you encounter contradiction. BUT can become very intensive within each considered option, there may be further points where this approach is required, very quickly leading to exponential runtimes!

Other possibilities:

* Store Sudoku as an array - computationally quicker?
  + Actually doesn't seem like a huge gain, as none of the common algebraic or matrix operators apply to solving sudoku it seems.
* As a list of strings or numbers - Possibly the easiest/ smallest for storage
  + How to store blanks
  + Perhaps both

My proposal:

Store sudoku’s as a class, allowing us to store both the dictionary and list form of the sudoku in one object, which can then be accessed by different functions depending on what they need. Include in all updating of the sudoku an update to both.