# Artificial Intelligence Nanodegree

## Introductory Project: Diagonal Sudoku Solver

# Question 1 (Naked Twins)

Q: How do we use constraint propagation to solve the naked twins problem?

A:

Let us take the example of the solution of the Sudoku puzzle with grid,

'2.............62....1....7...6..8...3...9...7...6..4...4....8....52.............3'

We modify the program to detect naked twins on every call to the naked\_twins() function. The naked twins it detects during the different calls to the naked\_twins() function are as follows:

[('B2', 'F6')]

[('B2', 'F6')]

[('B2', 'F6')]

[('B2', 'F6')]

[('H1', 'H6')]

[('B2', 'F6')]

[('H1', 'H6')]

[('B1', 'B9')]

[('B3', 'B4')]

[('H1', 'H6')]

[('F3', 'I3')]

[('D5', 'F5')]

[('A3', 'B3')]

[('B1', 'C2')]

[('D5', 'F5')]

[('B1', 'B9')]

[('B3', 'B4')]

[('H1', 'H6')]

[('I2', 'I7')]

[('A3', 'B3')]

[('D5', 'F5')]

[('B9', 'F9')]

[('A3', 'B3')]

[('D5', 'F5')]

[('H1', 'I2')]

[('G8', 'I7')]

[('B3', 'B4')]

[('D1', 'F1')]

[('A3', 'B3')]

[('D5', 'F5')]

[('A3', 'B3')]

[('D1', 'F1')]

[('D5', 'F5')]

[('D5', 'F5')]

[('D5', 'F5')]

The naked\_twins\_eliminator() function called by naked\_twins() function does the job of eliminating the 2 numbers which are stuck with the naked twins boxes from all of their peers because no other peer can claim these numbers anymore. This naked\_twins\_eliminator() function called from the naked\_twins() function in fact acts as a constraint propagation strategy where other boxes’ possible outcomes are updated due to the outcomes of some other box. And this helps reduce/solve the problem.

# Question 2 (Diagonal Sudoku)

Q: How do we use constraint propagation to solve the diagonal sudoku problem?

A:

Let us take the example of the solution of the Sudoku puzzle with grid,

'2.............62....1....7...6..8...3...9...7...6..4...4....8....52.............3'

We add a new type of unit called diag\_units

[['A1', 'B2', 'C3', 'D4', 'E5', 'F6', 'G7', 'H8', 'I9'], ['A9', 'B8', 'C7', 'D6', 'E5', 'F4', 'G3', 'H2', 'I1']]

generated using-

diagonal1 = [a[0]+a[1] for a in zip(rows, cols)]

diagonal2 = [a[0]+a[1] for a in zip(rows, cols[::-1])]

diag\_units = [diagonal1,diagonal2]

to the unitlist data structure as follows:

unitlist = row\_units + column\_units + square\_units + diag\_units

This leads to introduction of additional new peers for the boxes lying on the diagonal

For eg: see the peers for A1 (which lies on the diagonal) compared to A2 (which does not lie on the diagonal)

'A1': {'C3', 'A8', 'G7', 'A9', 'E5', 'I9', 'F6', 'G1', 'A2', 'E1', 'A4', 'A6', 'H8', 'B2', 'A7', 'D1', 'D4', 'H1', 'B3', 'C2', 'A5', 'A3', 'F1', 'C1', 'I1', 'B1'},

'A2': {'E2', 'C3', 'A8', 'A9', 'A1', 'F2', 'A4', 'A6', 'B2', 'A7', 'H2', 'G2', 'D2', 'C2', 'B3', 'A5', 'A3', 'C1', 'I2', 'B1'},

The reduce\_puzzle() function makes repeated calls to the eliminate() function whose job is to go through all the boxes, and whenever there is a box with a value, eliminate this value from the values of all its peers. This peer information can also made use of when eliminating naked twins (If u in naked\_twins() function iterates over unit\_list instead of old\_unitlist.i.e in the case of diagonal naked twins.) which is the case in our program.

But largely, the updated peer information for a diagonal Sudoku along with the eliminate() function are used to achieve constraint propagation which helps reduce/solve the problem.

### Install

This project requires \*\*Python 3\*\*.

We recommend students install [Anaconda](https://www.continuum.io/downloads), a pre-packaged Python distribution that contains all of the necessary libraries and software for this project.

Please try using the environment we provided in the Anaconda lesson of the Nanodegree.

##### Optional: Pygame

Optionally, you can also install pygame if you want to see your visualization. If you've followed our instructions for setting up our conda environment, you should be all set.

If not, please see how to download pygame [here](http://www.pygame.org/download.shtml).

### Code

\* `solution.py` - You'll fill this in as part of your solution.

\* `solution\_test.py` - Do not modify this. You can test your solution by running `python solution\_test.py`.

\* `PySudoku.py` - Do not modify this. This is code for visualizing your solution.

\* `visualize.py` - Do not modify this. This is code for visualizing your solution.

### Visualizing

To visualize your solution, please only assign values to the values\_dict using the ```assign\_values``` function provided in solution.py

### Submission

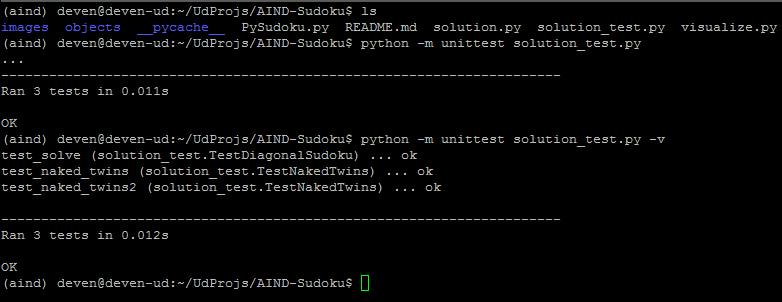
Before submitting your solution to a reviewer, you are required to submit your project to Udacity's Project Assistant, which will provide some initial feedback.

The setup is simple. If you have not installed the client tool already, then you may do so with the command `pip install udacity-pa`.

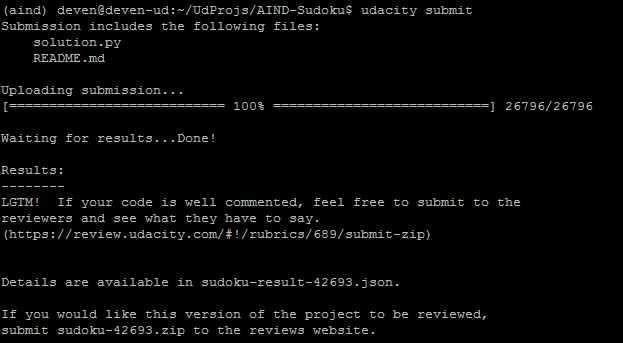
To submit your code to the project assistant, run `udacity submit` from within the top-level directory of this project. You will be prompted for a username and password. If you login using google or facebook, visit [this link](https://project-assistant.udacity.com/auth\_tokens/jwt\_login for alternate login instructions.

This process will create a zipfile in your top-level directory named sudoku-<id>.zip. This is the file that you should submit to the Udacity reviews system.

Running tests:



Submitting to Udacity submit pa:



It would be great to know from the reviewers about the following:

1. Ideas on how to make code cleaner, shorter would help. I don't know a lot of shortcuts in python and so have too many for loops and if loops.

2. Ideas/ Tools on debugging such programs with complex data structures which can become a mess quicky. I am using Pycharm currently but have just started using Pycharm.

3. I am new to Python and was trying to figure out a good way to find the runtime, memory usage etc. statistics for the program. Is there a way to include a few lines on the top and bottom in the program to do this or any other way?

4. Since Python is interpreted and does not do many optimizations like the way compiled languages do, what are they ways/best practices to achieve higher speed and low memory consumption

5. Any other things which might be a serious concern.