Section with Barbara

Week 1

Class Logistics

Knowledge

AI Stories

Grading

Projects

Friendly Reminder!

The projects are hard. Start them early ☺

Friendly Reminder!

Quizzes must be submitted on Gradescope

Friendly Reminder!

Double check your project submissions at submit.cs50.io

PROPOSITION

P

Not ¬
And ∧
Or ∨
Xor ⊕
Implication →
Biconditional ↔

Q

INFERENCE

Breast cancer 1-year survival rates are high. A patient has breast cancer. We can infer that a patient's 1-year survival rate is high.

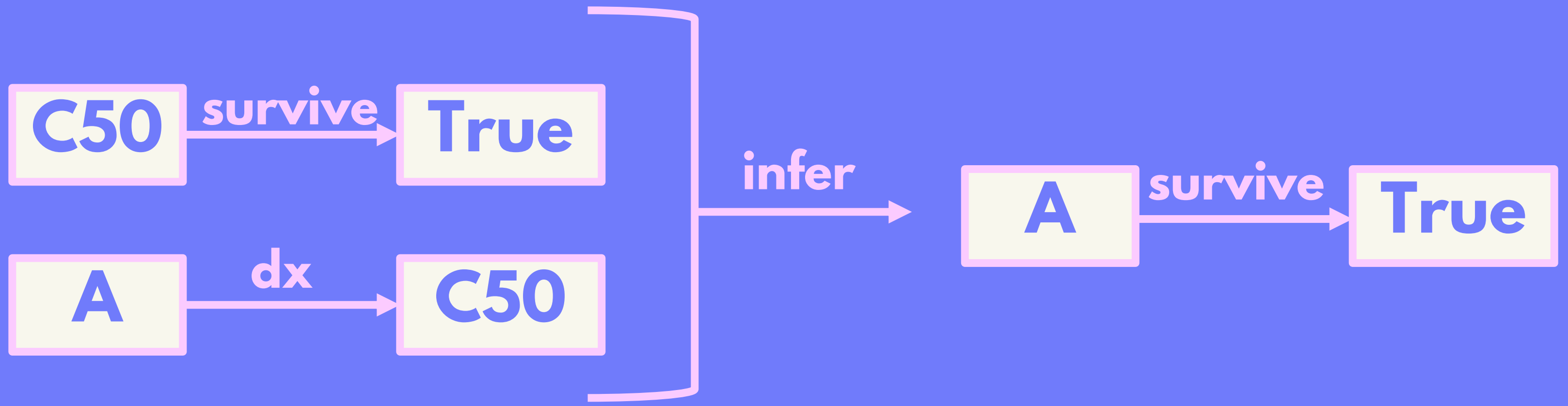
(an aside: Aristotle named this type of inference "BARBARA")

KNOWLEDGE BASE

person_id	survive
A	True
B	True
C	True

person_id	diagnosis
A	C50
B	C50
C	C50

database



knowledge base

ENTAILMENT

KB |= S
if all the propositions that evaluate to True in KB also evaluate to true in S

Model	survive	paclitaxel
C50	True	True
C34	False	True
C18	False	False

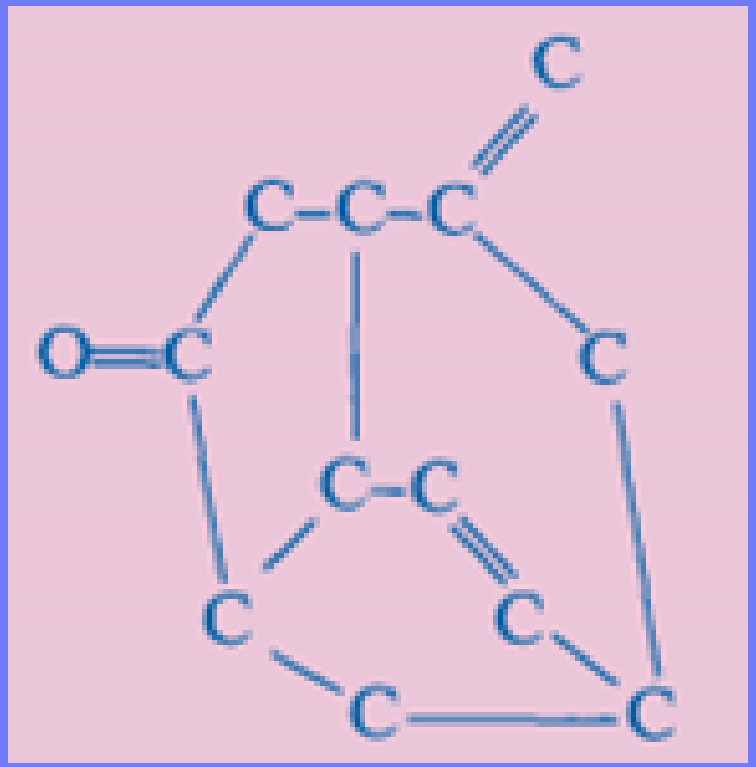
$$\begin{tikzcd}
C50 \arrow[r, "survive"] & True & & & & \\
& & {} \arrow[r, "infer"] & A \arrow[r, "survive"] & True \arrow[r, "paclitaxel"] & True \\
A \arrow[r, "dx"] & C50 & & & &
\end{tikzcd}$$

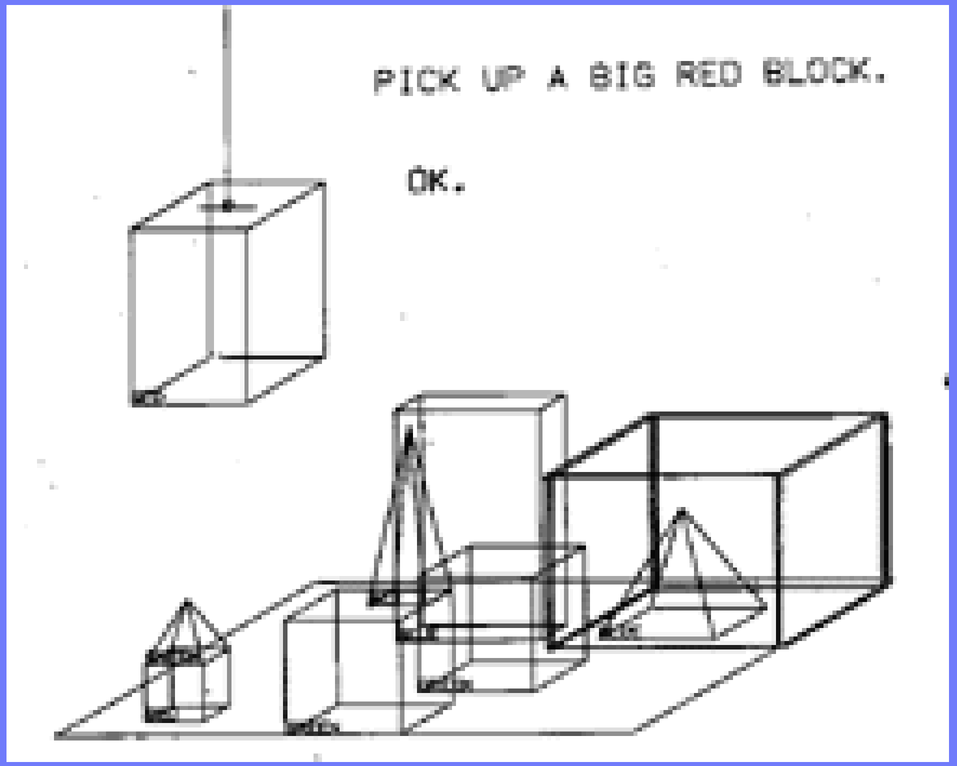
modus ponens

SEARCH + KNOWLEDGE

https://www.youtube.com/watch?v=mmQl6VGvX-c

AI Stories: DENDRAL, SHRDLU





GRADING - SUBMIT.CS50.IO

CORRECTNESS – CHECK50

STYLE – STYLE50

DESIGN – 🤷‍♀️ 🤷‍♂️

GOOD CODE DESIGN

Don't Repeat Yourself

Explain With Comments

Say It Concisely

Include All Scenarios

Group Common Cases

Name Objects Accurately

an anagram by Barbara

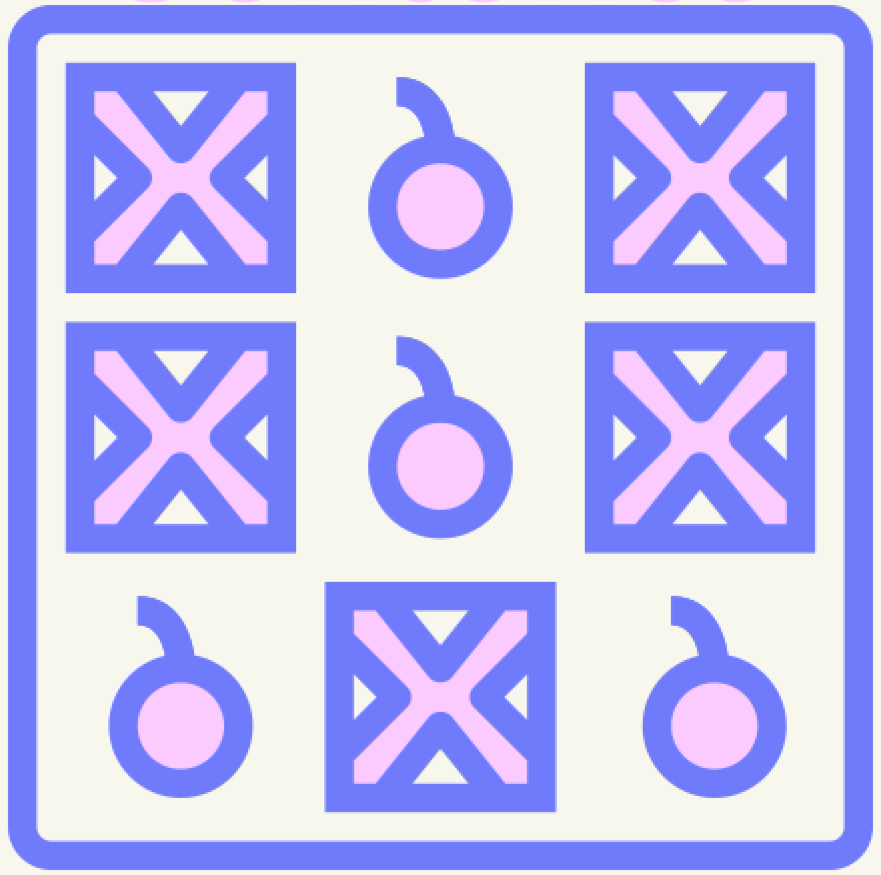
projects

WHO IS THE MURDERER?

- Three suspects have been brought in for a murder: Albert, George, and William.
- One of the three suspects is the murderer. The other two are innocent. Innocent suspects always tell the truth.
- Albert, George, and William all say that they are not the murderer.
- Albert additionally says that “William is the murderer”
- William additionally says that “Albert or George is innocent”

minesweeper

class Sentence



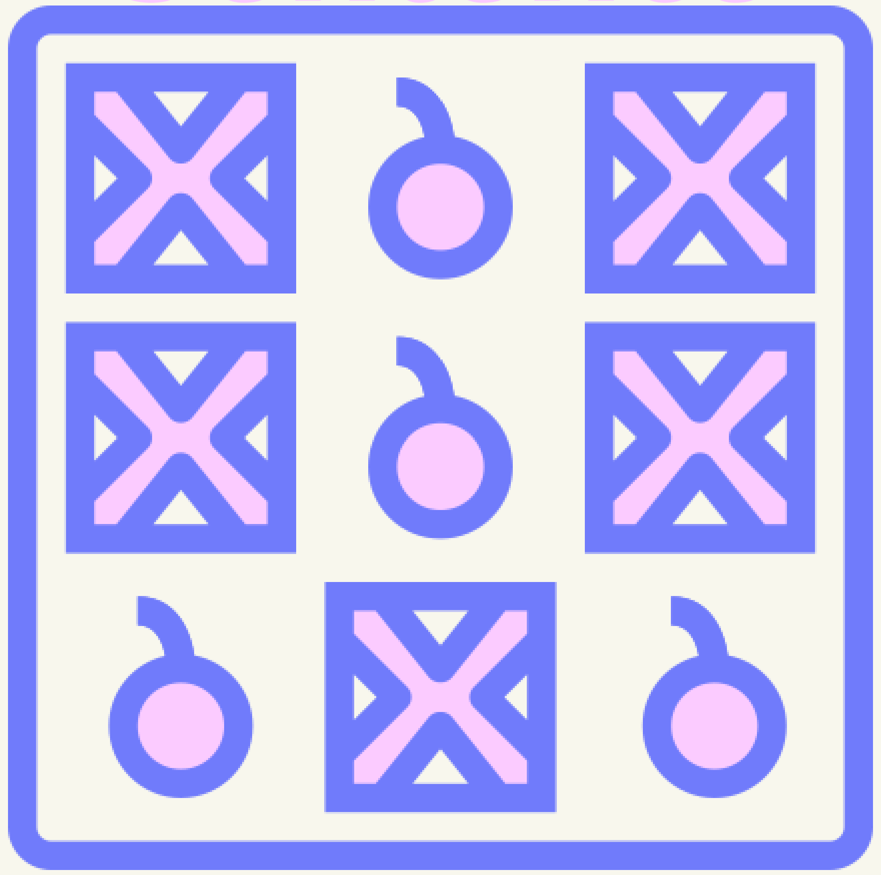
```
def known_mines(self):
        """
        Returns the set of all cells in self.cells known to be mines.
        """

def known_safes(self):
        """
        Returns the set of all cells in self.cells known to be safe.
        """
```

How can you use Sentence.cells and Sentence.count to help you implement these functions? What should be returned when there are no known safes or known mines in Sentence.cells?

minesweeper

class Sentence

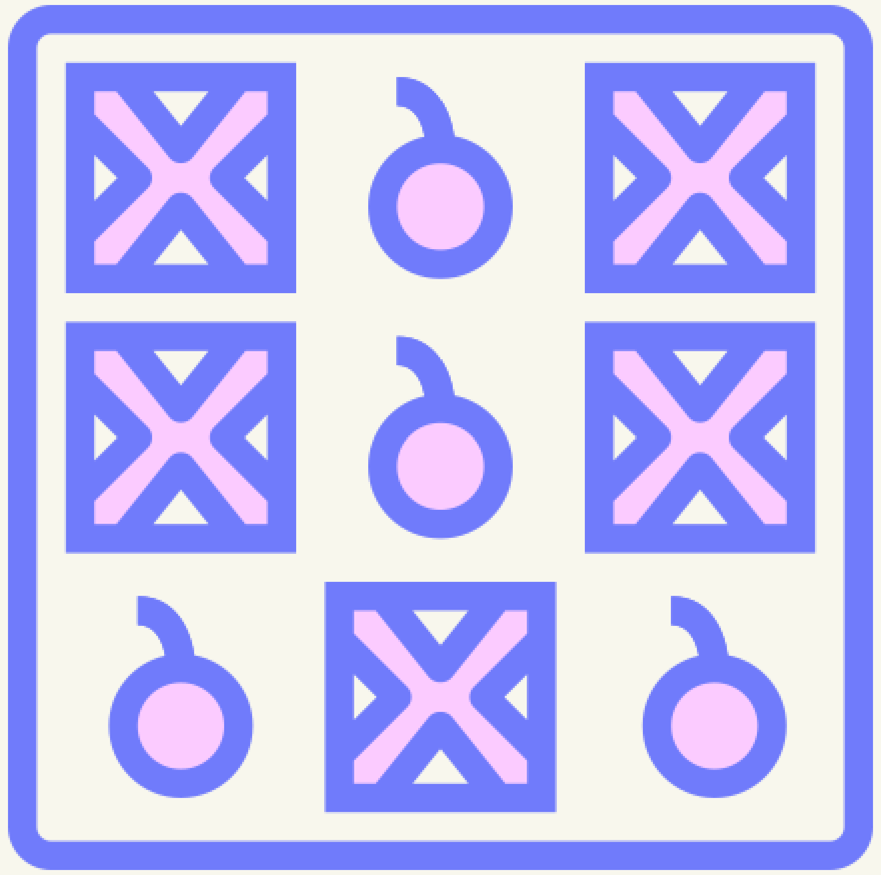


```
def mark_mine(self, cell):
       """
       Updates internal knowledge representation given the fact that
       a cell is known to be a mine.
       """
def mark_safe(self, cell):
       """
       Updates internal knowledge representation given the fact that
       a cell is known to be safe.
       """
```

How can you use Sentence.cells and Sentence.count to help you implement these functions? A cell should no longer be in Sentence if it is a known mine or known safe.

minesweeper

class MinesweeperAI

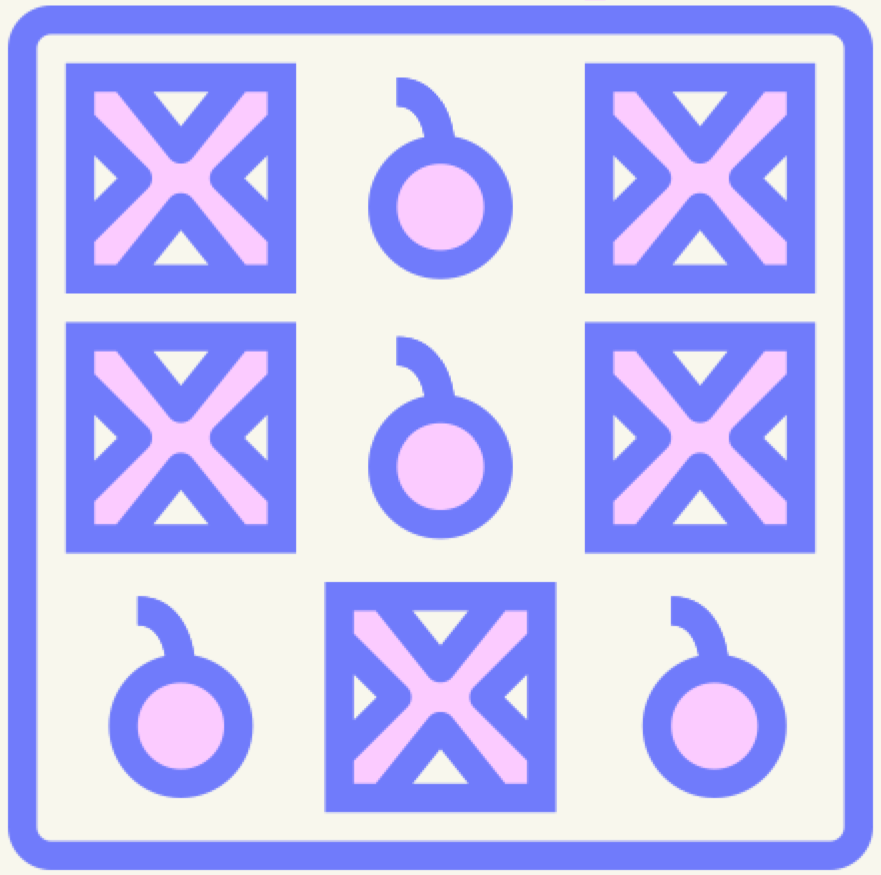


```
def add_knowledge(self, cell, count):
      """
      Called when the Minesweeper board tells us, for a given
      safe cell, how many neighboring cells have mines in them.
      This function should:
          1) mark the cell as a move that has been made
          2) mark the cell as safe
          3) add a new sentence to the AI's knowledge base
             based on the value of `cell` and `count`
          4) mark any additional cells as safe or as mines
             if it can be concluded based on the AI's knowledge
              base
          5) add any new sentences to the AI's knowledge base
             if they can be inferred from existing knowledge
      """
```

Knowing which attributes in MinesweeperAI from the project specifications to update is key! An extra challenge: how would you do this recursively?

minesweeper

class MinesweeperAI

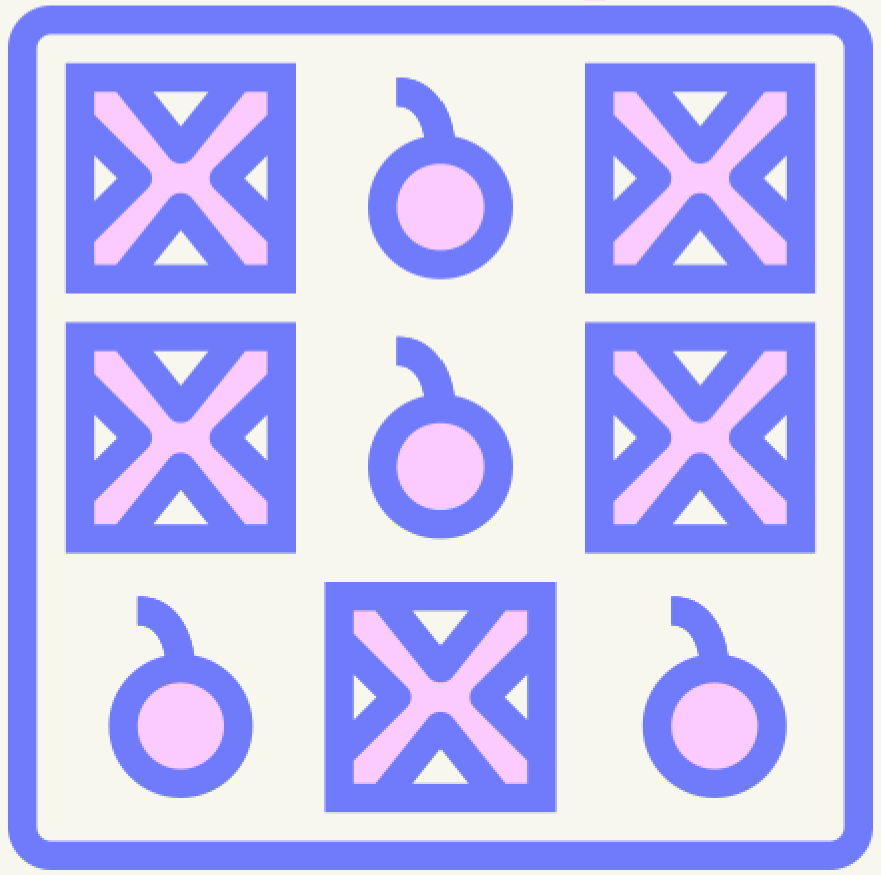


```
def make_safe_move(self):
        """
        Returns a safe cell to choose on the Minesweeper board.
        The move must be known to be safe, and not already a move
        that has been made.
        This function may use the knowledge in self.mines, self.safes
        and self.moves_made, but should not modify any of those
           values.
        """
```

If you know how to subtract a set from another set, this function will be easy!

minesweeper

class
MinesweeperAI



```
def make_random_move(self):
        """
        Returns a move to make on the Minesweeper board.
        Should choose randomly among cells that:
            1) have not already been chosen, and
            2) are not known to be mines
        """
```

random.choice will be helpful here!