**CS50's Introduction to Artificial Intelligence with Python**

***Lecture 0***

***BRIAN YU***

# [ARTIFICIAL INTELLIGENCE](https://cs50.harvard.edu/ai/2020/notes/0/#artificial-intelligence)

Artificial Intelligence (AI) covers a range of techniques that appear as sentient behavior by the computer. For example, AI is used to recognize faces in photographs on your social media, beat the World’s Champion in chess, and process your speech when you speak to Siri or Alexa on your phone.

In this course, we will explore some of the ideas that make AI possible:

1. **Search**

Finding a solution to a problem, like a navigator app that finds the best route from your origin to the destination, or like playing a game and figuring out the next move.

1. **Knowledge**

Representing information and drawing inferences from it.

1. **Uncertainty**

Dealing with uncertain events using probability.

1. **Optimization**

Finding not only a correct way to solve a problem, but a better—or the best—way to solve it.

1. **Learning**

Improving performance based on access to data and experience. For example, your email is able to distinguish spam from non-spam mail based on past experience.

1. **Neural Networks**

A program structure inspired by the human brain that is able to perform tasks effectively.

1. **Language**

Processing natural language, which is produced and understood by humans.

# [Knights](https://cs50.harvard.edu/ai/2020/notes/0/#search) Project

Write a program to solve logic puzzles.

## [Background](https://cs50.harvard.edu/ai/2020/projects/1/knights/#background)

In 1978, logician Raymond Smullyan published “What is the name of this book?”, a book of logical puzzles. Among the puzzles in the book were a class of puzzles that Smullyan called “Knights and Knaves” puzzles.

In a Knights and Knaves puzzle, the following information is given: Each character is either a knight or a knave. A knight will always tell the truth: if knight states a sentence, then that sentence is true. Conversely, a knave will always lie: if a knave states a sentence, then that sentence is false.

The objective of the puzzle is, given a set of sentences spoken by each of the characters, determine, for each character, whether that character is a knight or a knave.

For example, consider a simple puzzle with just a single character named A. A says “I am both a knight and a knave.”

Logically, we might reason that if A were a knight, then that sentence would have to be true. But we know that the sentence cannot possibly be true, because A cannot be both a knight and a knave – we know that each character is either a knight or a knave, but not both. So, we could conclude, A must be a knave.

That puzzle was on the simpler side. With more characters and more sentences, the puzzles can get trickier! Your task in this problem is to determine how to represent these puzzles using propositional logic, such that an AI running a model-checking algorithm could solve these puzzles for us.

## [Getting Started](https://cs50.harvard.edu/ai/2020/projects/1/knights/#getting-started)

* Download the distribution code from <https://cdn.cs50.net/ai/2020/x/projects/1/knights.zip> and unzip it.

## [Understanding](https://cs50.harvard.edu/ai/2020/projects/1/knights/#understanding)

Take a look at logic.py, which you may recall from Lecture 1. No need to understand everything in this file, but notice that this file defines several classes for different types of logical connectives. These classes can be composed within each other, so an expression like And(Not(A), Or(B, C)) represents the logical sentence stating that symbol A is not true, and that symbol B or symbol C is true (where “or” here refers to inclusive, not exclusive, or).

Recall that logic.py also contains a function model\_check. model\_check takes a knowledge base and a query. The knowledge base is a single logical sentence: if multiple logical sentences are known, they can be joined together in an And expression. model\_check recursively considers all possible models, and returns True if the knowledge base entails the query, and returns False otherwise.

Now, take a look at puzzle.py. At the top, we’ve defined six propositional symbols. AKnight, for example, represents the sentence that “A is a knight,” while AKnave represents the sentence that “A is a knave.” We’ve similarly defined propositional symbols for characters B and C as well.

What follows are four different knowledge bases, knowledge0, knowledge1, knowledge2, and knowledge3, which will contain the knowledge needed to deduce the solutions to the upcoming Puzzles 0, 1, 2, and 3, respectively. Notice that, for now, each of these knowledge bases is empty. That’s where you come in!

The main function of this puzzle.py loops over all puzzles, and uses model checking to compute, given the knowledge for that puzzle, whether each character is a knight or a knave, printing out any conclusions that the model checking algorithm is able to make.

## [Specification](https://cs50.harvard.edu/ai/2020/projects/1/knights/#specification)

An automated tool assists the staff in enforcing the constraints in the below specification. Your submission will fail if any of these are not handled properly, if you import modules other than those explicitly allowed, or if you modify functions other than as permitted.

Add knowledge to knowledge bases knowledge0, knowledge1, knowledge2, and knowledge3 to solve the following puzzles.

* Puzzle 0 is the puzzle from the Background. It contains a single character, A.
  + A says “I am both a knight and a knave.”
* Puzzle 1 has two characters: A and B.
  + A says “We are both knaves.”
  + B says nothing.
* Puzzle 2 has two characters: A and B.
  + A says “We are the same kind.”
  + B says “We are of different kinds.”
* Puzzle 3 has three characters: A, B, and C.
  + A says either “I am a knight.” or “I am a knave.”, but you don’t know which.
  + B says “A said ‘I am a knave.’”
  + B then says “C is a knave.”
  + C says “A is a knight.”

In each of the above puzzles, each character is either a knight or a knave. Every sentence spoken by a knight is true, and every sentence spoken by a knave is false.

Once you’ve completed the knowledge base for a problem, you should be able to run python puzzle.py to see the solution to the puzzle.

## [Hints](https://cs50.harvard.edu/ai/2020/projects/1/knights/#hints)

* For each knowledge base, you’ll likely want to encode two different types of information: (1) information about the structure of the problem itself (i.e., information given in the definition of a Knight and Knave puzzle), and (2) information about what the characters actually said.
* Consider what it means if a sentence is spoken by a character. Under what conditions is that sentence true? Under what conditions is that sentence false? How can you express that as a logical sentence?
* There are multiple possible knowledge bases for each puzzle that will compute the correct result. You should attempt to choose a knowledge base that offers the most direct translation of the information in the puzzle, rather than performing logical reasoning on your own. You should also consider what the most concise representation of the information in the puzzle would be.
  + For instance, for Puzzle 0, setting knowledge0 = AKnave would result in correct output, since through our own reasoning we know A must be a knave. But doing so would be against the spirit of this problem: the goal is to have your AI do the reasoning for you.
* You should not need to (nor should you) modify logic.py at all to complete this problem.

## [How to Submit](https://cs50.harvard.edu/ai/2020/projects/1/knights/#how-to-submit)

You may not have your code in your ai50/projects/2020/x/knights branch nested within any further subdirectories (such as a subdirectory called knights or project1a). That is to say, if the staff attempts to access https://github.com/me50/USERNAME/blob/ai50/projects/2020/x/knights/puzzle.py, where USERNAME is your GitHub username, that is exactly where your file should live. If your file is not at that location when the staff attempts to grade, your submission will fail.

1. Visit [this link](https://submit.cs50.io/invites/8f7fa48876984cda98a73ba53bcf01fd), log in with your GitHub account, and click **Authorize cs50**. Then, check the box indicating that you’d like to grant course staff access to your submissions, and click **Join course**.
2. [Install Git](https://git-scm.com/downloads) and, optionally, [install submit50](https://cs50.readthedocs.io/submit50/).
3. If you’ve installed submit50, execute
4. submit50 ai50/projects/2020/x/knights

Otherwise, using Git, push your work to https://github.com/me50/USERNAME.git, where USERNAME is your GitHub username, on a branch called ai50/projects/2020/x/knights.

1. Submit [this form](https://forms.cs50.io/a5e20f1f-c084-4ee3-ba85-95ca3422e338).

You can then go to <https://cs50.me/cs50ai> to view your current progress!

**IMPORTANT** note regarding your gradebook at [**cs50.me/cs50ai**](https://cs50.me/cs50ai)! Soon after the start of each calendar year (we estimate during the week of 11 January 2021), we typically refresh all of our gradebooks. All of your work from 2020 will be saved and will carry forward, but your gradebook may appear temporarily empty or unavailable until you have submitted work in 2021 that has been graded and returned to you by CS50 Bot. So don’t be alarmed!

If you finish CS50AI during the final days of December 2020, it is very important that you claim your free CS50 Certificate before this gradebook reset takes place, so don’t delay! (Verified certificates are unaffected by this.)