**[Understanding](https://cs50.harvard.edu/ai/2024/projects/0/degrees/" \l "understanding)**

The distribution code contains two sets of CSV data files: one set in the large directory and one set in the small directory. Each contains files with the same names, and the same structure, but small is a much smaller dataset for ease of testing and experimentation.

Each dataset consists of three CSV files. A CSV file, if unfamiliar, is just a way of organizing data in a text-based format: each row corresponds to one data entry, with commas in the row separating the values for that entry.

Open up small/people.csv. You’ll see that each person has a unique id, corresponding with their id in [IMDb](https://www.imdb.com/)’s database. They also have a name, and a birth year.

Next, open up small/movies.csv. You’ll see here that each movie also has a unique id, in addition to a title and the year in which the movie was released.

Now, open up small/stars.csv. This file establishes a relationship between the people in people.csv and the movies in movies.csv. Each row is a pair of a person\_id value and movie\_id value. The first row (ignoring the header), for example, states that the person with id 102 starred in the movie with id 104257. Checking that against people.csv and movies.csv, you’ll find that this line is saying that Kevin Bacon starred in the movie “A Few Good Men.”

Next, take a look at degrees.py. At the top, several data structures are defined to store information from the CSV files. The names dictionary is a way to look up a person by their name: it maps names to a set of corresponding ids (because it’s possible that multiple actors have the same name). The people dictionary maps each person’s id to another dictionary with values for the person’s name, birth year, and the set of all the movies they have starred in. And the movies dictionary maps each movie’s id to another dictionary with values for that movie’s title, release year, and the set of all the movie’s stars. The load\_data function loads data from the CSV files into these data structures.

The main function in this program first loads data into memory (the directory from which the data is loaded can be specified by a command-line argument). Then, the function prompts the user to type in two names. The person\_id\_for\_name function retrieves the id for any person (and handles prompting the user to clarify, in the event that multiple people have the same name). The function then calls the shortest\_path function to compute the shortest path between the two people, and prints out the path.

The shortest\_path function, however, is left unimplemented. That’s where you come in!

[**Specification**](https://cs50.harvard.edu/ai/2024/projects/0/degrees/#specification)

Complete the implementation of the shortest\_path function such that it returns the shortest path from the person with id source to the person with the id target.

* Assuming there is a path from the source to the target, your function should return a list, where each list item is the next (movie\_id, person\_id) pair in the path from the source to the target. Each pair should be a tuple of two strings.
  + For example, if the return value of shortest\_path were [(1, 2), (3, 4)], that would mean that the source starred in movie 1 with person 2, person 2 starred in movie 3 with person 4, and person 4 is the target.
* If there are multiple paths of minimum length from the source to the target, your function can return any of them.
* If there is no possible path between two actors, your function should return None.
* You may call the neighbors\_for\_person function, which accepts a person’s id as input, and returns a set of (movie\_id, person\_id) pairs for all people who starred in a movie with a given person.

You should not modify anything else in the file other than the shortest\_path function, though you may write additional functions and/or import other Python standard library modules.