Comparing my implementation with Gemini’s implementation.

1. The tools I used were from the standard Python library like random, math and to read the files I used open, readlines and split whereas Gemini used csv, random, math and DictReader.
2. My prompt to Gemini was the same from the spec and I told it to use basic code as I do not know complex and advanced functions. I also told it to set the minimum year as 1950 and max as 2120 but it did not do that correctly. I asked it to give me the same interaction window as the one given in spec.
3. There are a lot of differences between my code and Geminis’:
   - I am parsing through files manually using readlines and split. Gemini is using csv.DictReader - not really sure how that works. Although it looks like there is less room for error while using it.
   - I am using basic attributes for Person as in first_name, birth_rate but Gemini specifically mentioned that it will be using “protected” attributes like _first_name, _last_name, etc.
   - I am storing decades as int after converting it from string to int. Gemini is using decade keys as a string. I feel it would cause errors or get confusing.
   - My last name inheritance is not quite clear as I do not understand how to implement it correctly but I really liked how Gemini decides that children keep the last name of the biological descendant parent.
   - I only create a partner for the person if person.partner is none using the marriage rate whereas Gemini may create a partner during simulation for each person and will assign opposite gender partner only.
   - For duplicate name counting I use a dictionary where I store everyone and then I collect the names. Gemini goes through the whole list again and then uses .count() . My method is more efficient and avoids repeatedly scanning the list.
   - I use a recursive approach to build a tree whereas Gemini uses an iterative approach.
4. I would like to use csv.DictReader to read through files as it looks easier and shorter in code. It makes parsing safer. Also, I like the logic it uses for a child's last name, instead of picking randomly from the two root last names, the child consistently inherits last name from the biological parent in the tree. I felt that using an iterative approach for building a tree would be better when we have larger data sets and trees rather than using recursive loops.
5. I would not want to switch everything to getters/setters just because direct attributes felt totally reasonable in this assignment. I also liked my own implementation of collecting the duplicate names instead of counting them all in by going through the list again, it feels more complex. Additionally, I would not change my decade-based life expectancy lookup, using string can become complicated and ‘int’ will keep everything readable and simple.