Result Analysis:

Code quality

- **Gemini -** The code is straightforward and readable, but it relies on a lot of global variables and runs everything at the top level (no main()), which makes it a bit harder to maintain or extend cleanly. It has some clear easy to understand functions, but overall, it feels more like a quick script.
- **Cursor-** The structure is cleaner: it uses a main() function, has a docstring, and separates logic into small helpers like draw() and move_computer_paddle(). It also has extra game features (pause, mode selection) and the logic is more organized, so it feels more maintainable.

Speed of generation

- **Gemini -** Gemini got me to a working game very fast with minimal complexity, basic paddles, ball movement, and scoring came together quickly.
- **Cursor -** Cursor also generated a working version quickly, but it included more features, so it felt like a more complete build from the start.

Ease of use

- **Gemini -** Easy to follow because it's short and simple, but because everything is in one flow, making changes without breaking something can be less intuitive.
- **Cursor -** More user-friendly for development because it's broken into sections and functions. It's easier for me to locate what to edit and add features.

Debugging and error handling

- **Gemini -** It handles quitting properly, but debugging might take longer because most things are tied together, and it uses global state.
- **Cursor -**Easier to debug since the code is modular .The ball movement is also handled more predictably (using int() on movement), which helps reduce weird glitches.

Flexibility and customization

- **Gemini -** Good base for a simple Pong game, but adding features like menus, pause, difficulty levels, or two-player mode would require more refactoring.
- **Cursor -** Much easier to extend because it already supports multiple modes and has a better structure. I can add things like difficulty settings, sound, power-ups, or improved physics without rewriting the whole file.