Modeling the Problem

Players

Player 1 is the human. His/her assigned symbol is slash (`/`). Player 2 is the computer, and its symbol is backslash (`\`). Player 1 (human) starts first. Both players are created using Player object in `player.py` and stored inside the Board object in `board.py`. Players keep their points, name and symbol within them.

States

States are the status of the board after every single move. Current state is stored within the Board object defined in `board.py` with a variable name of cells (see `cells`). Variable `cells` is a `list[list[Cell]]` object representing a 2D array where each element is an instance of custom `Cell` class defined in `cell.py`. A cell keeps the symbol that was put in and the references to the neighboring circles.

Initial and Terminal States

Initial state consists of an empty board, 2 players with 0 points, a null previous state (later to be used for undoing, see `_save()` and `_recover()` functions in `board.py`). Terminal states are a collection of any state where there is no more empty cell to put a symbol (see `filled()` function in `board.py`).

State Transition Function

There are two state transition functions, `play_as_human()` and `play_as_computer()` in `board.py`, which take coordinates and apply corresponding modifications to the current state to generate the new state.

Flow

`main.py`

Starting point is `main.py`. User provides a text file, `board.txt`, to the program. This text file includes numbers to represent values of the circles. No circle is represented with 0 and treated as if there is a circle with a value of 0 in the corresponding corner. After that the main loop is initiated until the board is filled. In each iteration, program takes coordinate values from the user and registers corresponding symbol. Following that, board generates a response.

`board.py`

Function named `ai_respond()` is called. It creates an object, `SlantAI`, for all the possible actions from the current state. Each `SlantAI` object generates a score and `ai_respond()` picks the maximum among them and decides the correct move.

`slantai.py`

Takes the snapshot of the current state in initialization. Plays inherited move, the candidate move from `ai_respond()`, and calls the `_alphabeta()` function as a minimizer on the resulting state. The function `_alphabeta()` conducts a depth first search with alpha-beta pruning recursively to return a score. Heuristic function, `_h()`, is the points of the computer at the corresponding terminal state.

`main.py`

Returned move gets played with the function named `play_as_computer()`. Loop continues until the end and the winner is announced before termination.