A2 (CS10) - Code Documentation

The purpose of this program The purpose of this program is to create a set of tools to represent, manipulate, and analyze states of a Sliding Brick Puzzle. Below I go over the major components of the code.

`GameState` Class (`GameState.py`)

- This is the main class for managing the puzzle's state.
- Its primary attributes are the dimensions of the game grid (`rows` and `columns`), as well as the game grid itself (`grid`), which is represented as a 2D array.
- I wrote `__eq__` and `__hash__` functions to allow my `GameState` object to be put into a `set` object for the next part of the assignment.
- The `print` function prints the dimensions and the game grid to the command line.
- The `clone` function clones the `GameState` object and returns a new object.
- The `is_solved` function checks if the target shape (id=2) is covering the exit spaces (which are represented as -1 in the grid)
- The `compare` function compares two states and determines if their grids are equal.
- The `normalize` function takes into account the shapes of the blocks and not their shape ids and renumbers the shapes in ascending order starting from 3.
- The `GameState` class contains functions `_extract_shape_dictionary` and `_create_shapes`, which extracts all the different shape objects from the grid and creates a `Shape` object (see below) for each one. `GameState` maintains an attribute `shapes`, which is a dictionary that maps each shape `id` to its `Shape` object.

`Shape` Class (`Shape.py`)

- This class is a representation of an individual brick in a `GameState` grid.
- Its primary attributes are the shape's `id` and its `coordinates`. The minimum and maximum x and y coordinates are also stored to get the various faces of the shape.
- There are functions to get the top, bottom, left, and right faces of the shape.
- The `get_all_moves` uses the "check" functions in `utils.py`. There are check function for the four cardinal directions that takes as parameters a `Shape` and a `GameState.grid`, and checks whether a shape has space move in that direction on the grid. The `get_all_moves` function checks in all four directions and returns all of them that allow a possible move for the shape.
- The `make_move` function adjusts the `Shape`'s coordinates given a direction to move in.

Core Functionality (`GameState.py`)

- The `get_all_moves` function loops through every `Shape` in the grid and finds all the possible moves for each one. Then it outputs all the possible moves for every shape.
- The `apply_move` function modifies the game grid and the `Shape` objects in the grid to correspond to a new state after a move is made. There is also a `apply_move_clone` function that applies the move and returns a new cloned `GameState` object.
- The `random_walk` function takes in an initial state and a positive integer `N` as parameters. It will randomly make (available) moves on the game state until (i) the goal is reached, or (ii) the number of moves made in the random walk exceeds `N`.

Testing

Below are my outputs for the various functions in my `sbp.py` file, which is the entry point to my code.

`print`

```
sh run.sh print "SBP-level1.txt"
5, 5,
  1,  1,  1,  1,  1,
  1,  3,  2,  2,  1,
  1,  0,  4,  5,  1,
 -1,  0,  6,  7,  1,
  1,  1,  1,  1,  1,
```

```
sh run.sh print "SBP-level2.txt"
6, 5,
  1,  1,  1,  1,  1,  1,
  1,  0,  3,  2,  2,  1,
  1,  0,  3,  4,  5,  1,
 -1,  6,  6,  7,  8,  1,
```

`done`

```
sh run.sh done "SBP-level0.txt"
False
```

```
sh run.sh done "SBP-level0-solved.txt"
True
```

`availableMoves`

```
sh run.sh availableMoves "SBP-level1.txt"
(3, 'down')
(4, 'left')
(6, 'left')
```

`applyMove`

```
sh run.sh applyMove "SBP-level1.txt" "(3,down)"
5, 5,
  1,  1,  1,  1,  1,
  1,  0,  2,  2,  1,
  1,  3,  4,  5,  1,
 -1,  0,  6,  7,  1,
  1,  1,  1,  1,  1,
```

`compare`

```
sh run.sh compare "SBP-level0.txt" "SBP-level0-test.txt"
False
```

```
sh run.sh compare "SBP-level0.txt" "SBP-level0.txt"
True
```

`norm`

```
sh run.sh norm "SBP-test-not-normalized.txt"
6, 8,
  1,  1,  1,  1,  1,  1,
  1,  3,  2,  2,  4,  1,
  1,  5,  2,  2,  6,  1,
  1,  7,  7,  8,  8,  1,
  1,  9,  9, 10, 10,  1,
  1,  0,  0,  0,  0,  1,
  1,  0,  0,  0,  0,  1,
  1,  1, -1, -1,  1,  1,
```

`random`

```
sh run.sh random "SBP-level0.txt" 3
5, 4,
  1, -1, -1,  1,  1,
  1,  0,  3,  4,  1,
  1,  0,  2,  2,  1,
  1,  1,  1,  1,  1,


(3, 'up')
5, 4,
  1, -1,  3,  1,  1,
  1,  0,  0,  4,  1,
  1,  0,  2,  2,  1,
  1,  1,  1,  1,  1,


(2, 'left')
5, 4,
  1, -1,  3,  1,  1,
  1,  0,  0,  4,  1,
  1,  2,  2,  0,  1,
  1,  1,  1,  1,  1,


(2, 'right')
5, 4,
  1, -1,  3,  1,  1,
  1,  0,  0,  4,  1,
  1,  0,  2,  2,  1,
  1,  1,  1,  1,  1,
```