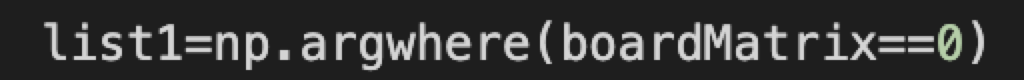
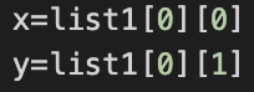
The design of Sliding Puzzle

1. Design (version 1.0)

1. Overview

The programmar use sys, random and numpy three modules and six functions to achieve the Sliding Puzzle Game.

1. Data Model
2. I mainly use  to track the sliding position. I use variables x and y to obtain the coordinate of the ‘0’..

c. Program Structure

1. I divided this game into six functions.

Function1 gameMode(): to receive the game mode 3-10 and convert it to next function.

Function2 board():to bulid a boardMatrix and convert it.

Function3 findZero():to obtain the sliding position and get the coordinate x,y.

Function4 draw\_board(): to eliminate zero and brackets.

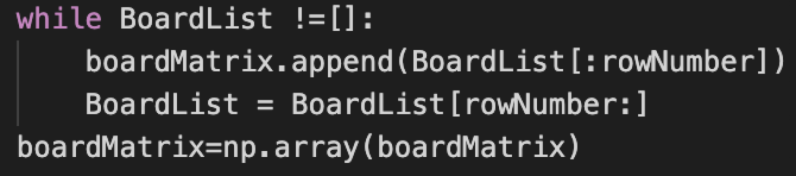
Function5 direction():to judge which position can ‘0’ move to.

Function6 judge():to judge whether the game win.

Function7 main():packing function into the main() function.

1. Processing Logic
2. First of all, to ensure the model of Sliding Puzzle, and convert the variable ‘mode’ to the next function. Then use the second function to build two Matrixs—boardMatrix and originMatrix(for the last step). Next I build the findZero() function to find the coordinate of ‘0’. I use the direction() function to make sure which position the ‘0’ can move to. I use the judeg() function and two lists to judge wheter the game succeed. Finally, I use main funciton to pack all functions and use ‘direct’ to achieve the movement of ‘0’.
3. Import numpy as np. Using the variable ‘mode’ to make sure the board is n\*n board. I use BoardList=[] to receive all numbers in a list. Then I use random.shuffle to randomize the puzzle.

Finally, I use



to transform the list to numpy.matix.

2. Function Specification

Function1 gameMode(): to receive the game mode 3-10 form users.

Only one parameter is ‘mode’. This function is designed to obtain the variable ‘mode’

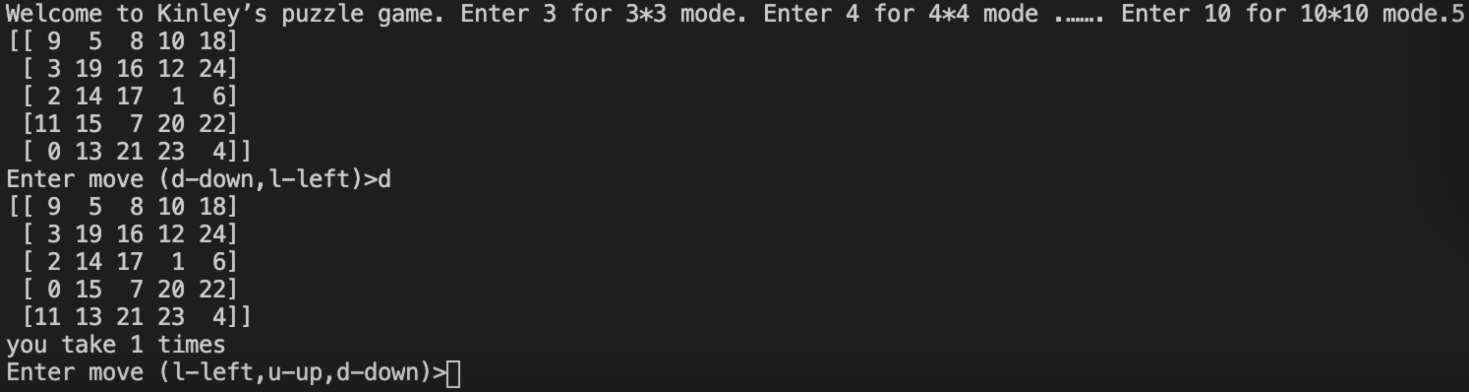
Function2 board(): Parameters include BoardList,OriginList,boardMatrix and originMatrix. The output is boardMatrix and originMatrix.

Function3 findZero():Parameters include x,y,list1. x and y is used to receive the coordinate of ‘0’

Function4 direction():to judge which position can ‘0’ move to.

Function5 judge():Parameters include list1 and list2. If list1==list2,game win, else, return False.

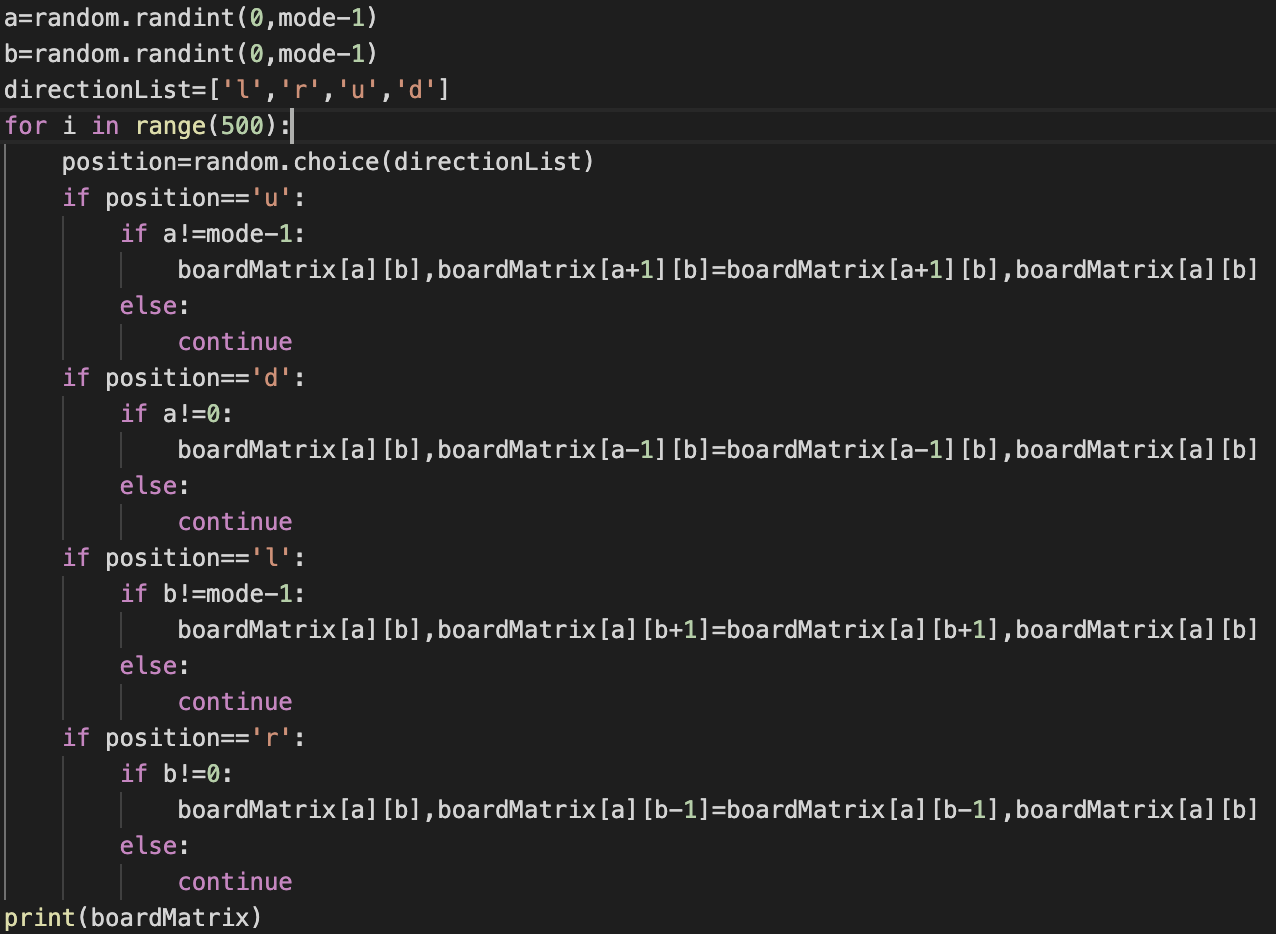
Function6 main():packing function into the main() function. Parameter includes ‘move’ and ‘direct’. ‘move’ is used to calculate the moves. ‘direct’ is used to change the position of ‘0’ sliding.

3.Output.

4.Further Revise. (version 2.0)

i.I use this later to replace randome.shuffle() function to reslove the problem that the board can not be restored.

In the new method, the way of randomizing the board likes Rubik's Cube. It uses exchanging the position of two numbers instead of casually randomizing it. Hence, it can be certainly restored!



ii. Introduce draw\_board() function to eliminate 0 and brackets.

