The 8 puzzle AI problem is a simple challenge of finding the correct combination of 'moves' from a given state to the final state in a display of 8 numbers on equal squares distributed in a 3x3 square-frame such that the end result is a progressive depiction of the numbers from 1-8 and the last box being empty (represented by 0).

Final state - [1, 2, 3]
[4, 5, 6]
[7, 8, 0]

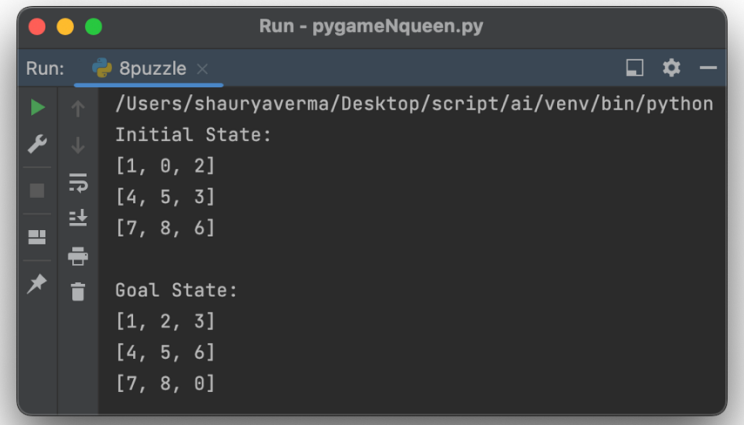
(In this code, the final state can also be changed based on user's preference)

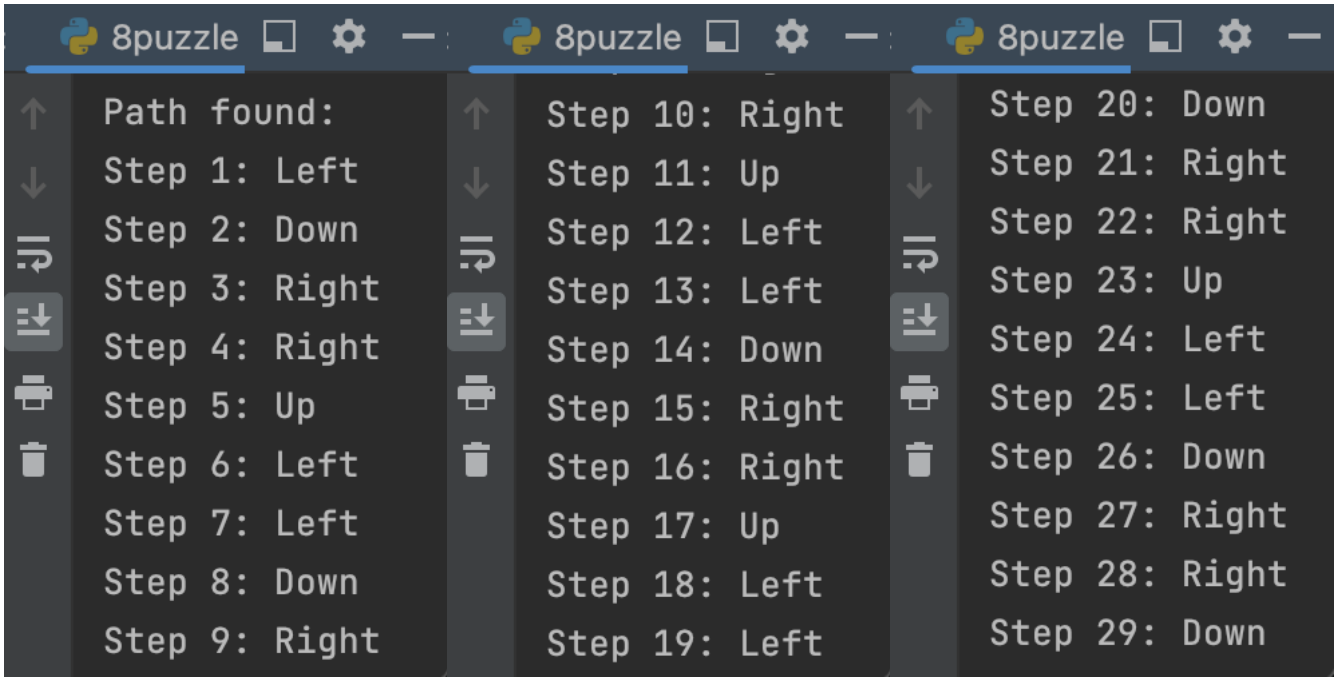
This uses DFS (Depth-First Search) to reach the solution which means the actions listed initially (left, right, up, down) define the result output. (Depicted with two sample action listings)

Example1.

```
# Actions: Up, Down, Left, Right
actions = [(0, -1), (0, 1), (-1, 0), (1, 0)]
action_labels = ['Left', 'Right', 'Up', 'Down']
```

OUTPUT –

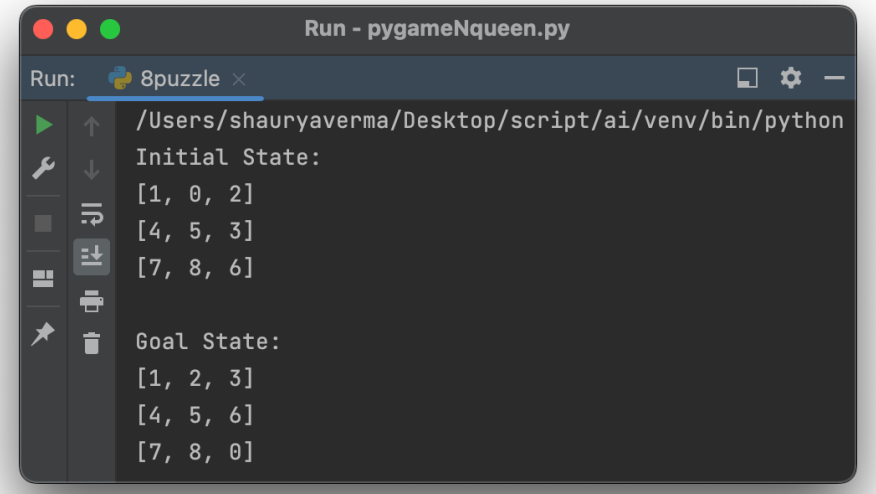


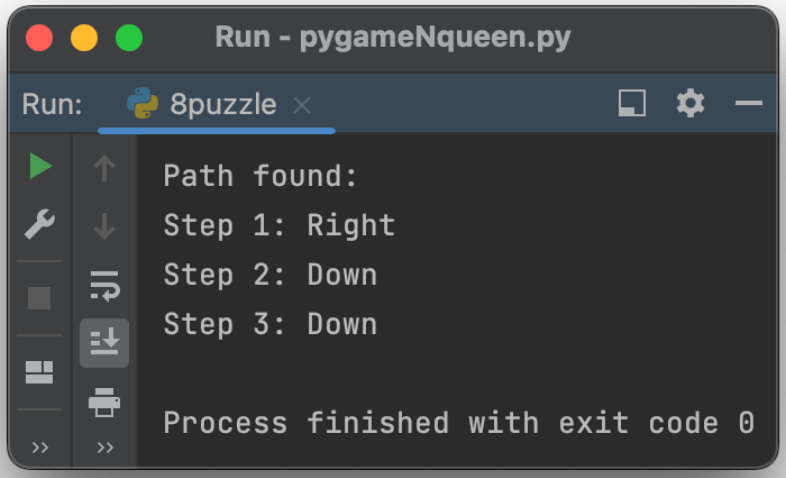


Example2.

```
# Actions: Up, Down, Left, Right
actions = [(0, 1), (1, 0), (-1, 0), (0, -1)]
action_labels = ['Right', 'Down', 'Up', 'Left']
```

OUTPUT –





Specific implementation of DFS algorithm takes place here to view the major fluctuations in output upon simple changes of actions.

Time complexity noted here depends on the change in actions provided.

The actions “Up, Left, Right, Down” are the directions of the “0” – empty square which replaces the neighbor square’s position with itself.

// checkout the pygame repository to have a graphical representation of the same.